aws

Guia do usuário

Amazon Neptune

Amazon Neptune: Guia do usuário

Table of Contents

O que é o Amazon Neptune?

O Amazon Neptune é um serviço de banco de dados de grafos rápido, confiável e totalmente gerenciado que facilita a criação e a execução de aplicações que trabalham com conjuntos de dados altamente conectados. O recurso principal do Neptune é um mecanismo de banco de dados de grafo com projeto específico e alto desempenho. Esse mecanismo é otimizado para armazenar bilhões de relacionamentos e consultar grafos com latência de milissegundos. O Neptune é compatível com as populares linguagens de consulta de grafos de propriedades Apache TinkerPop Gremlin e openCypher da Neo4j, e a linguagem de consulta RDF do W3C, SPARQL. Isso permite que você crie consultas que naveguem com eficiência por conjuntos de dados altamente conectados. O Neptune habilita casos de uso de grafos, como mecanismos de recomendação, detecção de fraudes, grafos de conhecimento, descoberta de medicamentos e segurança de rede.

O banco de dados Neptune é altamente disponível, com réplicas de leitura, recuperação para um ponto no tempo, backup contínuo para o Amazon S3 e replicação entre zonas de disponibilidade. O Neptune fornece atributos de segurança de dados, com suporte à criptografia em repouso e em trânsito. O Neptune é totalmente gerenciado, portanto, você não precisa mais se preocupar com tarefas de gerenciamento de banco de dados, como provisionamento de hardware, aplicação de patches no software, instalação, configuração ou backups.

O Neptune Analytics é um mecanismo de banco de dados analítico que complementa o banco de dados Neptune e pode analisar rapidamente grandes quantidades de dados de grafos na memória para obter insights e encontrar tendências. O Neptune Analytics é uma solução para analisar rapidamente bancos de dados de grafos existentes ou conjuntos de dados de grafos armazenados em um data lake. Ele usa algoritmos analíticos de grafos populares e consultas analíticas de baixa latência.

Para saber mais sobre o uso do Amazon Neptune, recomendamos começar com as seguintes seções:

- Conceitos básicos do Amazon Neptune
- Visão geral dos atributos do Amazon Neptune

Se você não tem conhecimento sobre grafos ou ainda não está com tudo pronto para investir em um ambiente de produção completo do Neptune, acesse nosso tópico Conceitos básicos para descobrir como usar os cadernos Jupyter no Neptune para aprendizado e desenvolvimento sem gerar custos.

Antes de começar a projetar um banco de dados, recomendamos consultar o repositório do GitHub AWS Reference Architectures for Using Graph Databases, onde você pode fundamentar suas escolhas sobre modelos de dados de grafos e linguagens de consulta, além de procurar exemplos de arquiteturas de implantação de referência.

Componentes principais do serviço

- Instância de banco de dados primária – oferece suporte a operações de leitura e gravação, além de realizar todas as modificações de dados no volume do cluster. Cada cluster de banco de dados do Neptune DB tem uma instância de banco de dados principal responsável por gravar (ou seja, carregar ou modificar) o conteúdo do banco de dados de grafos.
- Réplica do Neptune: conecta-se ao mesmo volume de armazenamento da instância de banco de dados principal e só é compatível com operações de leitura. Cada cluster de banco de dados do Neptune pode ter até 15 réplicas do Neptune, além da instância de banco de dados principal. Isso fornece alta disponibilidade ao localizar réplicas do Neptune em zonas de disponibilidade separadas e carga de distribuição dos clientes de leitura.
- Volume do cluster: os dados do Neptune são armazenados no volume do cluster, que é projetado para confiabilidade e alta disponibilidade. Um volume do cluster consiste em cópias dos dados em várias zonas de disponibilidade em uma única região da AWS. Como os dados são replicados automaticamente nas zonas de disponibilidade, eles são resilientes e há uma possibilidade pequena de perda de dados.

Compatível com APIs de gráficos abertas

O Amazon Neptune é compatível com APIs de grafos abertos para grafos de propriedades (Gremlin e openCypher) e grafos do RDF (SPARQL). Ele fornece alto desempenho para os modelos de gráfico e as linguagens de consulta. É possível escolher o modelo Property Graph (PG) e acessar o mesmo grafo com a linguagem de consulta openCypher e/ou a linguagem de consulta Gremlin. Se você usar o modelo Resource Description Framework (RDF) padrão do W3C, poderá acessar o grafo usando a linguagem de consulta SPARQL padrão.

Altamente seguro

O Neptune oferece vários níveis de segurança para o banco de dados. Os atributos de segurança incluem isolamento de rede usando a Amazon VPC e a criptografia em repouso usando chaves que você cria e controla por meio do AWS Key Management Service (AWS KMS). Em uma instância criptografada do Neptune, os dados do armazenamento subjacente são criptografados, bem como os backups, as réplicas e os snapshots automatizados no mesmo cluster.

Totalmente gerenciado

Com o Amazon Neptune, não é mais necessário se preocupar com tarefas de gerenciamento de banco de dados, como provisionamento de hardware, aplicação de patches em software, instalação, configuração ou backups.

É possível usar o Neptune para criar aplicações de grafos interativas e sofisticadas que podem consultar bilhões de relacionamentos em milissegundos. As consultas SQL de dados altamente conectados são complexas e o ajuste de seu desempenho é difícil. Com o Neptune, é possível usar as linguagens de consulta de grafos populares, Gremlin, openCypher e SPARQL, para executar consultas sofisticadas, fáceis de criar e com bom desempenho em dados conectados. Esse recurso reduz significativamente a complexidade de código para que você possa criar rapidamente aplicativos que processam relacionamentos.

O Neptune foi criado para oferecer disponibilidade superior a 99,99%. Ele aumenta o desempenho e a disponibilidade do banco de dados integrando totalmente o mecanismo de banco de dados a uma camada de armazenamento virtualizada com SSD criada especificamente para workloads de banco de dados. Armazenamento do Neptune é tolerante a falhas e com recuperação automática As falhas de disco são reparadas em segundo plano sem perda da disponibilidade do banco de dados. O Neptune detecta falhas no banco de dados e o reinicia automaticamente sem necessidade de recuperação de pane nem de recriar o cache do banco de dados. Se a instância inteira falhar, o Neptune realizará um failover automaticamente em uma das 15 réplicas de leitura.

Alterações e atualizações no Amazon Neptune

A tabela a seguir descreve alterações importantes no Amazon Neptune.

Alteração	Descrição	Data
Versão do motor 1.3.2.1	Em 2024-06-20, a versão 1.3.2.1 do mecanismo geralmente está sendo implantada. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine versão 1.3.2.1.	20 de junho de 2024
Versão do motor 1.3.2.0	Em 2024-06-10, a versão 1.3.2.0 do mecanismo geralmente está sendo implantada. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine versão 1.3.2.0.	10 de junho de 2024
Versão do motor 1.2.1.1	Em 11/03/2024/03, a versão 1.2.1.1 do mecanismo geralmente está sendo implantada. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para	11 de março de 2024

	obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine versão 1.2.1.1.	
Versão do motor 1.3.1.0	Em 2024-03-06, a versão 1.3.1.0 do motor está sendo geralmente implantada. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine versão 1.3.1.0.	6 de março de 2024
Atualização das permissões de políticas AWS gerenciadas	As políticas `NeptuneReadOnlyAcess` `NeptuneFullAccess` gerenciadas agora incluem `Sid` (ID da declaração) como um identificador na declaração de política.	22 de janeiro de 2024
O Neptune agora oferece armazenamento otimizado para E/S	Com o armazenamento otimizado para E/S, você paga pelo armazenamento e pelas instâncias que está usando. Os custos de armazenamento são mais altos do que os do armazenamento padrão, mas você não paga nada por qualquer E/S que você usa.	13 de dezembro de 2023

Alterações na política gerenciada do IAM para o Neptune	A política gerenciada NeptuneConsoleFullAccessdo IAM foi atualizada para conceder as permissões necessárias para interagir com os gráficos do Neptune Analytics, uma NeptuneGr aphReadOnlynova política de acesso foi adicionada para fornecer acesso somente de leitura aos recursos gráficos do Neptune Analytics e uma nova `AWSServic eRoleForNeptuneGra phPolicy` política foi adicionada para permitir que os gráficos do Neptune Analytics publiquem métricas e registros operacionais e de uso. CloudWatch	29 de novembro de 2023
Versão 1.3.0.0 do mecanismo	Desde 15/11/2023, a versão 1.3.0.0 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponíve l em todas as regiões. Para obter mais informações sobre essa versão do mecanismo , consulte Neptune Engine Release 1.3.0.0.	15 de novembro de 2023

O Neptune foi lançado na região de Israel (Tel Aviv)	O Amazon Neptune agora está disponível na região de Israel (Tel Aviv) (`il-central-1`).	13 de novembro de 2023
Postagem no blog sobre a implementação de time-to-live gráficos de propriedades em Neptune	Consulte Implement Time to Live in Amazon Neptune, Part 1: Property Graph, de Melissa Kwok, Mike Havey e Kevin Phillips.	27 de outubro de 2023
Versão 1.2.1.0.R7 do mecanismo	Desde 06/10/2023, a versão 1.2.1.0.R7 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre esta versão do mecanismo, consulte Neptune Engine Release 1.2.1.0.R7.	6 de outubro de 2023
Versão 1.2.0.0.R4 do mecanismo	Desde 29/09/2023, a versão 1.2.0.0.R4 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.2.0.0.R4.	29 de setembro de 2023

Versão 1.2.0.1.R3 do mecanismo	Desde 27/09/2023, a versão 1.2.0.1.R3 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.2.0.1.R3.	27 de setembro de 2023
Versão 1.2.1.0.R6 do mecanismo	Desde 12/09/2023, a versão 1.2.1.0.R6 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.2.1.0.R6.	12 de setembro de 2023
Versão 1.2.0.2.R6 do mecanismo	Desde 12/09/2023, a versão 1.2.0.2.R6 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.2.0.2.R6.	12 de setembro de 2023

Postagem no blog sobre o uso de uma estratégia de implantação azul/verde para atualizações do mecanismo do Neptune	Consulte Improve availability of Amazon Neptune during engine upgrade using blue/green deployment, de Ankit Gupta e Abhishek Mishra.	11 de setembro de 2023
Versão 1.2.1.0.R5 do mecanismo	Desde 02/09/2023, a versão 1.2.1.0.R5 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.2.1.0.R5.	2 de setembro de 2023
Versão 1.2.0.2.R5 do mecanismo	Desde 16/08/2023, a versão 1.2.0.2.R5 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.2.0.2.R5.	16 de agosto de 2023

Versão 1.2.1.0.R4 do mecanismo	Desde 10/08/2023, a versão 1.2.1.0.R4 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.2.1.0.R4.	10 de agosto de 2023
Postagem no blog sobre a versão 1.2.1.0 do mecanismo do Neptune	Consulte Exploring the feature packed 1.2.1.0 release for Amazon Neptune, de Joy Wang, Kevin Phillips, Andrea Nassisi e Navtanay Sinha.	4 de agosto de 2023
Postagem no blog sobre a criação de uma solução de banco de dados multimodal com o Neptune	Consulte Design a use case-driven, highly scalable multimodel database solution using Amazon Neptune, de Mike Havey.	18 de julho de 2023
Postagem no blog sobre os casos de uso e práticas recomendadas do Neptune Serverless	Consulte Use cases and best practices to optimize cost and performance with Amazon Neptune Serverless, de Kevin Phillips e Ankit Gupta.	28 de junho de 2023

Versão 1.2.1.0.R3 do mecanismo	Desde 13/06/2023, a versão 1.2.1.0.R3 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.2.1.0.R3.	13 de junho de 2023
Postagem no blog sobre como gerar sugestões de lazer em tempo real	Consulte Generate suggestions for leisure activities in real time with Amazon Neptune, de Michael Meidlinger e Nils Müller.	6 de junho de 2023
Postagem no blog sobre modelagem molecular com Neptune e RDKit	Consulte Model molecular SMILES data with Amazon Neptune and RDKit, de Graham Kutchek.	1.º de junho de 2023
Postagem no blog sobre como usar o cache para acelerar o desempenho do Neptune (Parte 3)	Consulte Acelere o desempenho de consultas gráficas com o armazenamento em cache no Amazon Neptune, Parte 3: arquiteturas de cache em todo o cluster do Neptune com a Amazon, de Taylor Riggan, Abhishek Mishra, Melissa Kwok e Kelvin ElastiCache Lawrence.	26 de maio de 2023

Postagem no blog sobre como usar o cache para acelerar o desempenho do Neptune (Parte 2)	Consulte Accelerate graph query performance with caching in Amazon Neptune, Part 2: Additional Neptune caching features, de Taylor Riggan, Abhishek Mishra, Melissa Kwok e Kelvin Lawrence.	26 de maio de 2023
Postagem no blog sobre como usar o cache para acelerar o desempenho do Neptune (Parte 1)	Consulte Accelerate graph query performance with caching in Amazon Neptune, Part 1: Queries and buffer pool caching, de Taylor Riggan, Abhishek Mishra, Melissa Kwok e Kelvin Lawrence.	26 de maio de 2023
Postagem no blog sobre análise da cadeia de suprimentos usando o Neptune	Consulte Supply chain data analysis and visualization using Amazon Neptune and the Neptune workbench, de Dhiraj Thakur e Rajdip Chaudhur.	10 de maio de 2023
Versão 1.2.0.2.R4 do mecanismo	Desde 08/05/2023, a versão 1.2.0.2.R4 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponíve l em todas as regiões. Para obter mais informações sobre essa versão do mecanismo , consulte Neptune Engine Release 1.2.0.2.R4.	8 de maio de 2023

Lançamento do Neptune na região do Oriente Médio (EAU)	O Amazon Neptune já está disponível na região do Oriente Médio (EAU) (`me-central-1`).	2 de maio de 2023
Versão 1.2.1.0.R2 do mecanismo	Desde 02/05/2023, a versão 1.2.1.0.R2 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.2.1.0.R2.	2 de maio de 2023
Postagem no blog sobre como criar um grafo de conhecimento sobre o Neptune com análise de vídeo baseada em IA usando o Media2Cloud	Consulte Build a knowledge graph on Amazon Neptune with AI-powered video analysis using Media2Cloud, de Mike Havey.	2 de maio de 2023
Postagem no blog sobre como criar uma plataforma DevOcean de remediação de vulnerabilidades usando o Neptune	Veja Como DevOcean criei uma plataforma de gerenciamento de remediação de vulnerabilidades para aplicativos nativos da nuvem usando o Amazon Neptune, de Gil Makmel e Charles Ivie.	25 de abril de 2023

Postagem no blog sobre como a Getir cria um sistema de detecção de fraudes usando o Neptune	Consulte How Getir build a comprehensive fraud detection system using Amazon Neptune and Amazon DynamoDB, de Berkay Berkman, Mahmut Turan, Mutlu Polatcan, Umut Cemal Kıraç, Yağız Yanıkoğlu e Esra Kayabali.	6 de abril de 2023
Postagem no blog sobre como a Wiz remodela a segurança na nuvem usando o Neptune	Consulte The World is a graph: How Wiz reimagines cloud security using a graph in Amazon Neptune, de Ami Luttwak e Brad Bebee.	31 de março de 2023
Versão 1.2.0.2.R3 do mecanismo	Desde 27/03/2023, a versão 1.2.0.2.R3 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.2.0.2.R3.	27 de março de 2023
Postagem no blog sobre como a CSC Generation potencializa a descoberta de produtos usando o Neptune	Consulte How CSC Generation powers product discovery with knowledge graphs using Amazon Neptune, de Bobber Cheng, Ronit Rudra e Melissa Kwok.	21 de março de 2023

Versão 1.2.1.0 do mecanismo	Desde 08/03/2023, a versão 1.2.1.0 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.2.1.0.	8 de março de 2023
Postagem no blog sobre como explorar os novos recursos do TinkerPop 3.6.x no Neptune	Consulte Explorando os novos recursos do Apache TinkerPop 3.6.x no Amazon Neptune, de Stephen Mallette.	8 de março de 2023
Postagem no blog sobre o uso do raciocínio semântico para inferir novos fatos do grafo RDF	Consulte Use semantic reasoning to infer new facts from your RDF graph by integrating RDFox with Amazon Neptune, de Charles Ivie e Diana Marks.	20 de fevereiro de 2023
Postagem no blog sobre a análise de dados FHIR de saúde com o Neptune	Consulte Analyze healthcare FHIR data with Amazon Neptune, de Alena Schmickl.	13 de fevereiro de 2023
Postagem no blog sobre como criar uma solução de detecção de fraudes em tempo real usando o Neptune ML	Consulte Build a real-time fraud detection solution using Amazon Neptune ML, de Hua Shu e Soji Adeshina.	8 de fevereiro de 2023

Versão 1.1.1.0.R7 do mecanismo	Desde 23/01/2023, a versão 1.1.1.0.R7 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.1.1.0.R7.	23 de janeiro de 2023
Lançamento do Graph-explorer	O graph-explorer é uma ferramenta de aplicação web de front-end de código aberto utilizada para visualizar dados de grafos. Consulte https://github.com/aws/graph-explorer.	3 de janeiro de 2023
Versão de manutenção 1.1.0.0.R3	Desde 23/12/2022, a versão de manutenção 1.1.0.0.R3 da versão do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.1.0.0.R3.	23 de dezembro de 2022

O Neptune Workbench agora opera no Amazon Linux 2 e 3. JupyterLab	Os notebooks gráficos Neptune agora são executados em um ambiente Amazon Linux 2 com 3. JupyterLab Consulte Migrando seus notebooks Neptune do Jupyter JupyterLab para o 3 para obter informações sobre como migrar para esse novo ambiente.	21 de dezembro de 2022
Postagem no blog sobre recomendações e pesquisas avançadas usando um grafo de conhecimento da IMDb (parte 3)	Consulte Power recommendations and search using an IMDb knowledge graph – Part 3, de Divya Bhargavi, Soji Adeshina, Gaurav Rele, Karan Sindwani, Vidya Sagar Ravipati e Matthew Rhodes.	20 de dezembro de 2022
Postagem no blog sobre recomendações e pesquisas avançadas usando um grafo de conhecimento da IMDb (parte 2)	Consulte Power recommendations and search using an IMDb knowledge graph – Part 2, de Matthew Rhodes, Soji Adeshina, Divya Bhargavi, Gaurav Rele, Karan Sindwani e Vidya Sagar Ravipati.	20 de dezembro de 2022
Postagem no blog sobre recomendações e pesquisas avançadas usando um grafo de conhecimento da IMDb (parte 1)	Consulte Power recommendations and search using an IMDb knowledge graph – Part 1, de Gaurav Rele, Soji Adeshina, Divya Bhargavi, Karan Sindwani, Vidya Sagar Ravipati e Matthew Rhodes.	20 de dezembro de 2022

O Neptune Serverless agora está disponível em novas regiões AWS	Em 16/12/2022, o Neptune Serverless foi lançado nas seguintes regiões da AWS : Canadá (Central), Europa (Estocolmo), Europa (Frankfurt), Ásia-Pacífico (Singapura) e Ásia-Pacífico (Sydney). Consulte Amazon Neptune Serverless constraints para conhecer todas as regiões onde o Neptune Serverless está disponível.	16 de dezembro de 2022
Versão 1.2.0.2.R2 do mecanismo	Desde 15/12/2022, a versão 1.2.0.2.R2 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.2.0.2.R2.	15 de dezembro de 2022
Versão 1.2.0.0.R3 do mecanismo	Desde 15/12/2022, a versão 1.2.0.0.R3 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.2.0.0.R3.	15 de dezembro de 2022

Versão 1.2.0.1.R2 do mecanismo	Desde 13/12/2022, a versão 1.2.0.1.R2 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.2.0.1.R2.	13 de dezembro de 2022
Postagem no blog sobre como projetar uma arquitetura educacional de análise de big data com AWS	Consulte Designing an educational big data analysis architecture with AWS, de Lavanya Sood.	13 de dezembro de 2022
Postagem no blog sobre como bancos de dados de grafos podem aprimorar o aprendizado	Consulte How graph databases can enhance learning, de Lavanya Sood.	08 de dezembro de 2022
Postagem no blog sobre como carregar dados RDF no Neptune usando o Glue AWS	Consulte Carregar dados RDF no Amazon AWS Neptune com Glue, de Mike Havey e Fabrizio Napolitano.	23 de novembro de 2022

Versão 1.2.0.2 do mecanismo	Desde 20/11/2022, a versão 1.2.0.2 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.2.0.2.	20 de novembro de 2022
Versão 1.2.0.1 do mecanismo	Desde 26/10/2022, a versão 1.2.0.1 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.2.0.1.	26 de outubro de 2022
Postagem no blog sobre detecção de fraudes usando o Neptune	Consulte Empowering fraud detection at Delivery Hero with Amazon Neptune, de Wilson Tang, Amr Elnaggar, Matias Pons, Mohammad Azzam, Saurabh Deshpande e Luis Rodrigues Soares.	26 de outubro de 2022
Postagem no blog sobre o Neptune Serverless	Consulte Introducing Amazon Neptune Serverless – A Fully Managed Graph Database that Adjusts Capacity for Your Workloads, de Danilo Poccia.	26 de outubro de 2022

Postagem no blog sobre importações de RDF orientadas por eventos no Neptune usando Lambda e SPARQL UPDATE LOAD	Veja como a NXP executa importações de RDF orientadas por eventos para o Amazon Neptune usando o AWS Lambda e o SPARQL UPDATE LOAD de John Walker, Onno Buijs, Charles Ivie e Javy de Koning.	20 de outubro de 2022
Versão 1.2.0.0.R2 do mecanismo	Desde 14/10/2022, a versão 1.2.0.0.R2 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.2.0.0.R2.	14 de outubro de 2022
Postagem no blog sobre codificação de propriedades de texto multilíngue no Neptune	Consulte Encode multi-lingual text properties in Amazon Neptune to train predictive models, de Jiani Zhang.	14 de outubro de 2022
Postagem no blog sobre testes automatizados do acesso a dados do Neptune	Consulte Teste automatizado do acesso aos dados do Amazon Neptune com o TinkerPop Apache Gremlin de Greg Biegel.	28 de setembro de 2022

Versão 1.1.1.0.R6 do mecanismo	Desde 23/09/2022, a versão 1.1.1.0.R6 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.1.1.0.R6.	23 de setembro de 2022
Postagem no blog sobre como a Informatica® usa o Neptune	Consulte How Informatica® Cloud Data Governance and Catalog uses Amazon Neptune for knowledge graphs, de Tiju Titus John, Deepak Ram e Farooq Ashraf.	20 de setembro de 2022
Postagem no blog sobre o uso do Neptune e de perspectivas da Tom Sawyer para descobrir fraudes financeiras	Consulte Uncover financial fraud with Amazon Neptune and Tom Sawyer Perspectives, de Janet M. Six, gerente sênior de produtos da Tom Sawyer Software.	30 de agosto de 2022
Postagem no blog sobre a criação de uma solução de detecção de fraudes em tempo real baseada em GNN usando SageMaker Neptune e DGL	Consulte Crie uma solução de detecção de fraudes em tempo real baseada em GNN usando a Amazon SageMaker, o Amazon Neptune e a Deep Graph Library de Jian Zhang, Haozhu Wang e Mengxin Zhu.	11 de agosto de 2022

Postagem no blog sobre o uso de tags de recursos para interromper e iniciar os recursos do ambiente Neptune	Consulte Automate the stopping and starting of Amazon Neptune environment resources using resource tags, de Kevin Phillips.	1º de agosto de 2022
Postagem no blog sobre um artista que contribui para o Apache TinkerPop	Veja Beyond Code: The Artist Who Contributes to Apache, de TinkerPop Stephen Mallette e Ketrina Thompson.	1º de agosto de 2022
Postagem no blog sobre controle de acesso minucioso para ações do plano de dados do Neptune	Consulte Fine Grained Access Control for Amazon Neptune data plane actions, de Abhishek Mishra e Ankit Gupta.	29 de julho de 2022
Postagem no blog sobre bancos de dados globais do Neptune	Consulte Introducing Amazon Neptune Global Database, de Navtanay Sinha.	27 de julho de 2022
Versão 1.2.0.0 do mecanismo	Desde 21/07/2022, a versão 1.2.0.0 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.2.0.0.	21 de julho de 2022

Versão 1.1.1.0.R5 do mecanismo	Desde 21/07/2022, a versão 1.1.1.0.R5 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.1.1.0.R5.	21 de julho de 2022
Versão 1.1.1.0.R4 do mecanismo	Desde 23/06/2022, a versão 1.1.1.0.R4 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.1.1.0.R4.	23 de junho de 2022
Fluxos de trabalho simplificados de análise de grafos e de machine learning com integração Python	Agora é possível realizar tarefas de análise de grafos e de machine learning em dados de grafos armazenados no Amazon Neptune por meio de uma integração Python de código aberto que simplifica a ciência de dados e os fluxos de trabalho de ML. Consulte a documentação do AWS Data Wrangler para Neptune.	7 de junho de 2022

Versão 1.1.1.0.R3 do mecanismo	Desde 07/06/2022, a versão 1.1.1.0.R3 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.1.1.0.R3.	7 de junho de 2022
Postagem no blog sobre detecção de notícias falsas	Consulte Detect social media fake news using graph machine learning with Amazon Neptune ML, de Hasan Shojaei e Sarita Joshi.	19 de maio de 2022
Postagem no blog sobre o uso do SQL Server Integration Services (SSIS) com o Neptune	Consulte Discover new insights from your data using SQL Server Integration Services (SSIS) and Amazon Neptune, de Mesgana Gormley e Melissa Kwok.	18 de maio de 2022
Versão de manutenção 1.1.1.0.R2	Desde 16/05/2022, a versão de manutenção 1.1.1.0.R2 da versão do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.1.1.0.R2.	16 de maio de 2022

Versão de manutenção 1.1.0.0.R2	Desde 16/05/2022, a versão de manutenção 1.1.0.0.R2 da versão do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.1.0.0.R2.	16 de maio de 2022
Versão de manutenção 1.0.5.1.R4	Desde 16/05/2022, a versão de manutenção 1.0.5.1.R4 da versão do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.0.5.1.R4.	16 de maio de 2022

Versão de manutenção 1.0.5.0.R5	Desde 16/05/2022, a versão de manutenção 1.0.5.0.R5 da versão do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.0.5.0.R5.	16 de maio de 2022
Postagem no blog sobre a disponibilidade geral do openCypher no Neptune	Consulte Announcing the General Availability of openCypher support for Amazon Neptune, de Navtanay Sinha e Dave Bechberger.	22 de abril de 2022
Versão 1.1.1.0 do mecanismo	Desde 19/04/2022, a versão 1.1.1.0 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.1.1.0.	19 de abril de 2022

Postagem no blog sobre linhagem de dados	Consulte Criar linhagem de dados para data lakes usando AWS Glue, Amazon Neptune e Spline, de Khoa Nguyen, Krithivasan Balasubramaniyan e Rahul Shaurya.	1.º de abril de 2022
As atualizações para a versão 1.1.0.0 foram reativadas.	Em 21/02/2022, as atualizaç ões para a versão 1.1.0.0 do mecanismo foram temporari amente desativadas. Elas já foram reativadas.	22 de março de 2022
Postagem no blog sobre confiabilidade de projetos de serviços públicos	Consulte Using cloud-bas ed, data-informed, power system models to engineer utility reliability, de Abhineet Parchure.	22 de março de 2022
Lançamento do Neptune na África (Cidade do Cabo)	O Amazon Neptune agora está disponível na África (Cidade do Cabo) (`af-south-1`). No entanto, o suporte ao caderno do Neptune Workbench está temporariamente desativado no console do Neptune nessa região.	24 de fevereiro de 2022
Postagem no blog sobre grafos orientados por modelos usando OWL	Consulte Model-driven graphs using OWL in Amazon Neptune, de Mike Havey.	23 de fevereiro de 2022

Postagem no blog sobre como explorar grafos de conhecimento semântico com o Rhizomer	Consulte Explore the semantic knowledge graphs without SPARQL using Amazon Neptune with Rhizomer, de Roberto García.	22 de fevereiro de 2022
Postagem no blog sobre como representar graficamente a rede elétrica	Veja Representando graficamente a rede elétrica AWS, de Bobby Wilson e Joseph Beer.	18 de fevereiro de 2022
Novas opções de codificação de atributos de texto do Neptune ML	O Neptune agora FastText suporta a codificação de texto Sentence BERT para treinamento. Veja os FastTextrecursos do Neptune ML e os recursos do Sentence BERT no Neptune ML.	15 de fevereiro de 2022
Postagem no blog sobre consultas geoespaciais usando OpenSearch o Neptune	Consulte Combine o Amazon Neptune e o OpenSearch Amazon Service para consultas geoespaciais, de Ross Gabay e Abhilash Vinod.	1º de fevereiro de 2022
Postagem no blog sobre descoberta de crimes financeiros usando o Amazon EKS e o Neptune	Consulte Financial Crime Discovery using Amazon EKS and Graph Databases, de Severin Gassauer-Fleissner e Zahi Ben Shabat.	1º de fevereiro de 2022

O volume de um cluster do Neptune agora pode se expandir para até 128 tebibytes (TiB)	Em todas as regiões suportadas, exceto na China e GovCloud, o limite de tamanho de um volume de cluster do Neptune agora aumentou de 64 TiB para 128 TiB. Isso se aplica a todas as versões do mecanismo a partir da versão 1.0.2.2. Consulte a página de armazenamento do Amazon Neptune.	1º de fevereiro de 2022
A pesquisa de texto completo do Neptune agora se integra a todas as versões do. OpenSearch	Consulte Pesquisa de texto completo no Amazon Neptune usando OpenSearch o Amazon Service.	28 de janeiro de 2022
Postagem no blog sobre o uso de contêineres Docker para implantar cadernos gráficos	Consulte Usar contêineres do Docker para implantar Graph Notebooks em AWS, de Ganesh Sawhney e Qiang Zhang.	22 de janeiro de 2022
Versão 1.0.5.1.R3 do mecanismo	Desde 13/01/2022, a versão 1.0.5.1.R3 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponíve l em todas as regiões. Para obter mais informações sobre essa versão do mecanismo , consulte Neptune Engine Release 1.0.5.1.R3.	13 de janeiro de 2022

Postagem no blog sobre um sistema de recomendação baseado em grafos usando o Neptune ML	Consulte Graph-based recommendation system with Neptune ML: An illustration on social network link prediction challenges, de Yanwei Cui e Will Badr.	12 de janeiro de 2022
Postagem no blog sobre ajuste de escala automático do Neptune	Consulte Auto scale your Amazon Neptune database to meet workload demands, de Navtanay Sinha e Sudhanshu Gupta.	29 de novembro de 2021
Postagem no blog sobre análises e visualizações interativas de dados de grafos	Consulte Crie análises e visualizações interativas de dados gráficos usando o Amazon Neptune, o Amazon Athena Federated Query e o Amazon, de Sandeep Veldi e QuickSight Abhishek Mishra.	24 de novembro de 2021
Postagem no blog sobre a detecção de fraudes de identidade usando aprendiza do profundo baseado em grafos	Consulte How Careem is detecting identity fraud using graph-based deep learning and Amazon Neptune, de Kevin O'Brien, Kamran Habib e Will Badr.	23 de novembro de 2021

Versão 1.1.0.0 do mecanismo	Desde 19/11/2021, a versão 1.1.0.0 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.1.0.0.	19 de novembro de 2021
Postagem no blog sobre centralização da proteção e conformidade de dados	Consulte Centralização da proteção e conformidade de dados no Amazon AWS Neptune with Backup, de Brian O'Keefe.	8 de novembro de 2021
Postagem no blog sobre combate a fraudes e pagamentos indevidos	Consulte Fighting fraud and improper payments in real-time at the scale of federal expenditures, de Vladi Royzman e Spencer Smith.	2 de novembro de 2021
Postagem no blog sobre prevenção de cadastros falsos em contas	Consulte Prevent fake account sign-ups in real time with AI using Amazon Fraud Detector, de Anjan Biswas.	29 de outubro de 2021

Versão 1.0.5.1.R2 do mecanismo	Desde 26/10/2021, a versão 1.0.5.1.R2 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.0.5.1.R2.	26 de outubro de 2021
Postagem no blog sobre como prever o risco de embarcações em HawkEye 360 graus usando a Deep Graph Library	O See HawkEye 360 prevê o risco de embarcações usando a Deep Graph Library e o Amazon Neptune de Tim Pavlick, Ian Avilez, Dan Ford e Gaurav Rele.	15 de outubro de 2021
Versão 1.0.5.1 do mecanismo	Desde 01/10/2021, a versão 1.0.5.1 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.0.5.1.	1.º de outubro de 2021

Postagem no blog sobre por que os desenvolvedores gostam TinkerPop	Veja Por que os desenvolvedores gostam do Apache TinkerPop, uma estrutura de código aberto para computação gráfica, de Brad Bebee, Kelvin Lawrence e Stephen Mallette.	27 de setembro de 2021
Versão 1.0.5.0.R3 do mecanismo	Desde 15/09/2021, a versão 1.0.5.0.R3 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.0.5.0.R3.	15 de setembro de 2021
Postagem no blog sobre como criar uma solução de descoberta de dados com o Amundsen e o Neptune	Consulte Building a data discovery solution with Amundsen and Amazon Neptune, de Peter Hanssens e Don Simpson.	8 de setembro de 2021
O Neptune atualizou o pesquisador de fluxos para aceitar consultas de pesquisa de texto completo sem strings.	Nesta versão estão incluídas muitas melhorias na pesquisa de texto completo, incluindo compatibilidade com indexação de valores de propriedades que não sejam strings. Consulte OpenSearch Indexação sem seqüências de caracteres no Amazon Neptune.	23 de agosto de 2021

Versão 1.0.5.0.R2 do mecanismo	Desde 16/08/2021, a versão 1.0.5.0.R2 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.0.5.0.R2.	16 de agosto de 2021
Versão 1.0.4.2.R5 do mecanismo	Desde 16/08/2021, a versão 1.0.4.2.R5 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.0.4.2.R5.	16 de agosto de 2021
Postagem no blog sobre o suporte ao protocolo Graph Store no Neptune	Consulte Introducing Graph Store Protocol support for Amazon Neptune, de Chris Smith.	2 de agosto de 2021
Postagem no blog sobre os novos atributos do Neptune ML	Consulte Discover more insights in your graphs with new features from Amazon Neptune ML, de Soji Adeshina.	30 de julho de 2021

Postagem no blog sobre como obter previsões mais rapidamente com o Neptune ML	Consulte Get predictions for evolving graph data faster with Amazon Neptune ML, de Soji Adeshina.	30 de julho de 2021
Postagem no blog sobre machine learning de grafos mais fácil e rápido com o Neptune ML	Consulte Easier and faster graph machine learning with Amazon Neptune ML, de Soji Adeshina.	30 de julho de 2021
Postagem no blog sobre o suporte do Neptune openCypher	Consulte Announcing openCypher for Amazon Neptune: Building better graph applications with openCypher and Gremlin together, de Brad Bebee.	29 de julho de 2021
Versão 1.0.5.0 do mecanismo	Desde 27/07/2021, a versão 1.0.5.0 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.0.5.0.	27 de julho de 2021

Versão 1.0.4.2.R4 do mecanismo	Desde 23/07/2021, a versão 1.0.4.2.R4 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.0.4.2.R4.	23 de julho de 2021
Lançamento do Neptune na China (Pequim)	O Amazon Neptune agora está disponível na China (Pequim) (`cn-north-1`).	21 de julho de 2021
Versão 1.0.4.2.R3 do mecanismo	Desde 28/06/2021, a versão 1.0.4.2.R3 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.0.4.2.R3.	28 de junho de 2021
Postagem no blog sobre como a Dream11 escalou a rede social usando o Neptune	Veja Saiba como a Dream11, a maior plataforma de esportes de fantasia do mundo, expande sua rede social com o Amazon Neptune e a Amazon. ElastiCache	25 de junho de 2021

Publicação no blog sobre como transformar dados em conhecimento com o PoolParty Neptune e o Semantic Suite	Consulte Transforme dados em conhecimento com o PoolParty Semantic Suite e o Amazon Neptune, de Ioanna Lytra e Albin Ahmeti.	16 de junho de 2021
Postagem no blog sobre o uso do Neptune para explorar a base de conhecimento UniProt	Consulte Explorando a base de conhecimento sobre UniProt proteínas com AWS Open Data e Amazon Neptune, de Eric Greene, Rafa Xu e Yuan Shi.	10 de junho de 2021
Postagem no blog sobre o uso do Neptune para análise de risco baseada em dados	Veja as notas de campo: Análise de risco baseada em dados com o Amazon Neptune e o OpenSearch Amazon Service, de Adriaan de Jonge e Rohit Satyanarayana.	10 de junho de 2021
Versão 1.0.4.2.R2 do mecanismo	Desde 01/06/2021, a versão 1.0.4.2.R2 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.0.4.2.R2.	1º de junho de 2021
Postagem no blog sobre como usar o Neptune para visualizar sua infraestrutura AWS	Consulte Visualize sua AWS infraestrutura com o Amazon Neptune e o Config, de AWS Rohan Raizada e Amey Dhavle.	25 de maio de 2021

Postagem no blog sobre configuração para usar o Data Lens com o Neptune	Consulte Configurar AWS serviços para criar um gráfico de conhecimento no Amazon Neptune usando o Data Lens, de Russell Waterson.	5 de maio de 2021
Postagem no blog sobre como criar um grafo de conhecimento no Neptune usando o Data Lens	Consulte Build a knowledge graph in Amazon Neptune using Data Lens, de Russell Waterson.	5 de maio de 2021
As versões 1.0.1.0, 1.0.1.1 e 1.0.1.2 do mecanismo agora estão obsoletas	A partir de agora, nenhuma nova instância de banco de dados será criada usando qualquer uma dessas versões do mecanismo ou quaisquer patches relacionados a elas.	26 de abril de 2021
Tradução do inglês do estudo de caso sobre o Ministério da Economia, Comércio e Indústria do Japão usando o Neptune	Consulte Japan's Ministry of Economy, Trade and Industry Powers gBizINFO Corporate Information Search Database with AWS.	31 de março de 2021
Postagem no blog sobre o uso do Neptune com o Amazon Comprehend e o Lex	Consulte Supercharge your knowledge graph using Amazon Neptune, Amazon Comprehend, and Amazon Lex, de Dave Bechberger.	31 de março de 2021
Postagem no blog sobre o uso de funções do Lambda com o Neptune	Consulte Usar funções do AWS Lambda com o Amazon Neptune, de Ian Robinson.	26 de março de 2021

Versão 1.0.4.1.R1.1 do mecanismo	Desde 22/03/2021, a versão 1.0.4.1.R1.1 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.0.4.1.R1.1.	22 de março de 2021
Versão 1.0.4.1.R2.1 do mecanismo	Desde 11/03/2021, a versão 1.0.4.1.R2.1 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.0.4.1.R2.1.	11 de março de 2021
Postagem no blog sobre o uso do caderno de grafos de código aberto do Neptune para visualização de grafos	Consulte Getting started with open source graph notebook for graph visualization, de Joy Wang, Ora Lassila e Stephen Mallette.	10 de março de 2021
Tutorial sobre a integração do Neptune com o mecanismo de descoberta de dados e de metadados da Amundsen	Consulte How to use Amundsen with Amazon Neptune, de Andrew Ciambrone.	2 de março de 2021

Versão 1.0.4.1.R2 do mecanismo	Desde 24/02/2021, a versão 1.0.4.1.R2 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.0.4.1.R2.	24 de fevereiro de 2021
Versão 1.0.4.0.R2 do mecanismo	Desde 24/02/2021, a versão 1.0.4.0.R2 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.0.4.0.R2.	24 de fevereiro de 2021
Versão 1.0.3.0.R3 do mecanismo	Desde 19/02/2021, a versão 1.0.3.0.R3 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.0.3.0.R3.	19 de fevereiro de 2021

Versão 1.0.2.2.R6 do mecanismo	Desde 19/02/2021, a versão 1.0.2.2.R6 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.0.2.2.R6.	19 de fevereiro de 2021
Postagem no blog sobre a criação de um grafo de conhecimento usando eventos do Amazon Comprehend	Consulte Building a knowledge graph in Amazon Neptune using Amazon Comprehend Events, de Brian O'Keefe, Graham Horwood e Navtanay Sinha.	19 de janeiro de 2021
Postagem no blog sobre como habilitar aplicações de dados gráficos com pouco código	Consulte Enabling low code graph data apps with Amazon Neptune and Graphistry, de Leo Meyerovich, Dave Bechberger e Taylor Riggan.	18 de janeiro de 2021
Adição de documentação ao caderno sobre conceitos básicos dos dados gráficos.	Adição de uma seção de integração com a bancada de trabalho do Neptune que ajuda você a começar a criar dados gráficos e desenvolver aplicações gráficas sem precisar criar um cluster do Neptune até que você esteja pronto.	15 de janeiro de 2021

Postagem no blog sobre como redefinir os dados gráficos do Neptune em segundos	Consulte Resetting your graph data in Amazon Neptune in seconds, de Niraj Jetly e Navtanay Sinha.	17 de dezembro de 2020
Publicação no blog sobre como a Novartis AG usa SageMaker Neptune com BERT	Veja, a Novartis AG usa a Amazon e o Amazon Neptune para criar SageMaker e enriquecer um gráfico de conhecimento usando o BERT, de Othmane Hamzaoui, Fatema Alkhanaizi e Viktor Malesevic.	14 de dezembro de 2020
Postagem no blog sobre a criação de um grafo de conhecimento com redes de tópicos	Consulte Building a knowledge graph with topic networks in Amazon Neptune, de Edward Brown, diretor de projetos de IA, Eduardo Piairo, arquiteto , Marcia Oliveira, cientista de dados chefe, e Jack Hampson, CEO da Deeper Insights.	14 de dezembro de 2020
Versão 1.0.4.1 do mecanismo	Desde 08/12/2020, a versão 1.0.4.1 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponíve l em todas as regiões. Para obter mais informações sobre essa versão do mecanismo , consulte Neptune Engine Release 1.0.4.1.	8 de dezembro de 2020

Postagem no blog sobre como começar a usar o Neptune ML	Consulte How to get started with Neptune ML, de George Karypis, Dave Bechberger e Karthik Bharathy.	8 de dezembro de 2020
O Neptune agora tem uma API de redefinição rápida.	Usando a API de redefinição rápida, é possível excluir de forma rápida e fácil todos os dados em um cluster de banco de dados. Consulte API de redefinição rápida.	4 de dezembro de 2020
Postagem no blog sobre a criação de um grafo de conhecimento biológico no Pendulum	Consulte Building a biological knowledge graph at Pendulum using Amazon Neptune, de Connor Skennerton.	26 de novembro de 2020
Postagem no blog sobre os novos recursos do TinkerPop 3.4.8 no Neptune	Consulte Explorando os novos recursos do Apache TinkerPop 3.4.8 no Amazon Neptune, de Stephen Mallette.	18 de novembro de 2020
Postagem no blog sobre o uso do serviço de pesquisa Amazon Kendra com o Neptune	Consulte Incorporating your enterprise knowledge graph into Amazon Kendra, de Yazdan Shirvany, Mohit Mehta e Dipto Chakravarty.	17 de novembro de 2020
Notificações de eventos agora disponíveis	O Neptune agora é compatível com notificações de eventos que você pode usar para monitorar clusters de banco de dados com maior facilidade. Consulte Using Neptune Event Notification, de.	29 de outubro de 2020

Endpoints personalizados agora disponíveis	O Neptune agora aceita endpoints personalizados para maior controle na conexão com instâncias de banco de dados. Consulte Connecting to Amazon Neptune Endpoints.	29 de outubro de 2020
Postagem no blog sobre o uso do AWS Database Migration Service (DMS) para preencher seu gráfico do Neptune	Consulte Preenchendo seu gráfico no Amazon Neptune a partir de um banco de dados relacional usando o Database AWS Migration Service (DMS) — Parte 4: Juntando tudo, de Chris Smith.	22 de outubro de 2020
Postagem no blog sobre o uso do AWS Database Migration Service (DMS) para preencher seu gráfico do Neptune	Consulte Preenchendo seu gráfico no Amazon Neptune a partir de um banco de dados relacional usando o Database AWS Migration Service (DMS) — Parte 3: Projetando o modelo RDF, de Chris Smith.	22 de outubro de 2020
Postagem no blog sobre o uso do AWS Database Migration Service (DMS) para preencher seu gráfico do Neptune	Consulte Preenchendo seu gráfico no Amazon Neptune a partir de um banco de dados relacional usando o Database AWS Migration Service (DMS) — Parte 2: Projetando o modelo gráfico de proprieda des, de Chris Smith.	22 de outubro de 2020

Postagem no blog sobre o uso do AWS Database Migration Service (DMS) para preencher seu gráfico do Neptune	Consulte Preenchendo seu gráfico no Amazon Neptune a partir de um banco de dados relacional usando o Database AWS Migration Service (DMS) — Parte 1: Configurando o cenário, de Chris Smith.	22 de outubro de 2020
Versão 1.0.4.0 do mecanismo	Desde 12/10/2020, a versão 1.0.4.0 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.0.4.0.	12 de outubro de 2020
Versão 1.0.3.0.R2 do mecanismo	Desde 12/10/2020, a versão 1.0.3.0.R2 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.0.3.0.R2.	12 de outubro de 2020

Versão 1.0.2.2.R5 do mecanismo	Desde 12/10/2020, a versão 1.0.2.2.R5 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.0.2.2.R5.	12 de outubro de 2020
Postagem no blog sobre como configurar a VPC para consultas federadas SPARQL	Consulte Configure Amazon VPC for SPARQL 1,1 Federated Query with Amazon Neptune, de Charles Ivie.	12 de outubro de 2020
Postagem no blog sobre como redigir uma exclusão em cascata do SPARQL	Consulte Write a cascading delete in SPARQL, de Ora Lassila.	5 de outubro de 2020
Postagem no blog sobre AWS recursos gráficos usando o Neptune	Consulte Representar graficamente seus AWS recursos com o Amazon Neptune, de Dave Bechberger.	28 de setembro de 2020
Postagem no blog sobre a criação do mapeamento terminológico MedDRA para farmacovigilância e notificação de eventos adversos usando o Neptune	Consulte Building Amazon Neptune based MedDRA terminology mapping for pharmacovigilance and adverse event reporting, de Vaijayanti Joshi, Deven Atnoor, Ph.D e Sudhanshu Malhotra.	24 de setembro de 2020

Postagem no blog sobre como criar um grafo de conhecimento a partir de um data warehouse usando o Neptune, para complementar a inteligência comercial	Consulte Complement Commercial Intelligence by Building a Knowledge Graph out of a Data Warehouse with Amazon Neptune, de Shahria Hossain e Mikael Graindorge.	23 de setembro de 2020
Postagem no blog sobre balanceamento de carga usando o cliente Neptune Gremlin	Consulte Load balance graph queries using the Amazon Neptune Gremlin Client, de Ian Robinson.	16 de setembro de 2020
Postagem no blog sobre personalização digital usando um grafo de identidade na Cox Automotive	Consulte Cox Automotive scales digital personalization using an identity graph powered by Amazon Neptune, de Carlos Rendon e Niraj Jetly.	16 de setembro de 2020
Postagem no blog sobre filtragem colaborativa em dados do Yelp	Consulte Using collaborative filtering on Yelp data to build a recommendation system in Amazon Neptune, de Chad Tindel.	8 de setembro de 2020
Postagem no blog sobre visualização de resultados de consultas no Amazon Neptune	Consulte Visualize query results using the Amazon Neptune workbench, de Kelvin Lawrence.	2 de setembro de 2020

O Neptune lançou a visualização gráfica.	O Amazon Neptune agora oferece amplos recursos de visualização gráfica em cadernos Jupyter na bancada de trabalho Neptune, junto com vários novos recursos que facilitam o uso dos cadernos. Consulte Visualização gráfica.	12 de agosto de 2020
Lançamento do Neptune na América do Sul (São Paulo)	O Amazon Neptune já está disponível na América do Sul (São Paulo) (`sa-east-1`).	6 de agosto de 2020
Lançamento do Neptune na Ásia-Pacífico (Hong Kong)	O Amazon Neptune já está disponível na Ásia-Pacífico (Hong Kong) (`ap-east-1`).	6 de agosto de 2020
Versão 1.0.3.0 do mecanismo	Desde 03/08/2020, a versão 1.0.3.0 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.0.3.0.	3 de agosto de 2020

Versão 1.0.2.2.R4 do mecanismo	Desde 23/07/2020, a versão 1.0.2.2.R4 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.0.2.2.R4.	23 de julho de 2020
Postagem no blog sobre o rastreamento automatizado de contatos da Zerobase usando o Amazon Neptune	Consulte Zerobase creates private, secure, and automated contact tracing using Amazon Neptune, de David Harris e Aron Szanto.	13 de julho de 2020
Lançamento do Neptune no Oeste dos EUA (Norte da Califórnia)	O Amazon Neptune já está disponível na Oeste dos EUA (Norte da Califórnia) (`us-west-1`).	9 de julho de 2020
O Amazon Neptune é compatível com o controle de acesso baseado em tags.	Agora você pode usar AWS tags nas políticas do IAM para controlar o acesso ao seu banco de dados Neptune. Consulte Tag-based access control in Amazon Neptune.	7 de julho de 2020

Um pesquisador de fluxos Java já está disponível.	O Amazon Neptune agora é compatível com uma versão Java do pesquisador de fluxos do Lambda para fluxos do Neptune, bem como para o Python. Consulte Add details about the Neptune streams consumer stack you are creating.	6 de julho de 2020
Publicação no blog sobre o gráfico de conhecimento da AWS COVID-19	Consulte Construindo e consultando o gráfico de conhecimento da AWS COVID-19, de Ninad Kulkarni, Colby Wise, George Price e Miguel Romero.	1º de julho de 2020
Versão 1.0.1.1 do mecanismo	Desde 26/06/2020, a versão 1.0.1.1 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponíve l em todas as regiões. Para obter mais informações sobre essa versão do mecanismo , consulte Neptune Engine Release 1.0.1.1.	26 de junho de 2020
Postagem no blog sobre como migrar do Blazegraph para o Amazon Neptune	Consulte Moving to the cloud: Migrating Blazegraph to Amazon Neptune, por Dave Bechberger.	25 de junho de 2020

Postagem no blog sobre como alterar a captura de dados do Neo4j para o Amazon Neptune	Consulte Change data capture from Neo4j to Amazon Neptune using Amazon Managed Streaming for Apache Kafka, por Sanjeet Sahay.	22 de junho de 2020
Postagem no blog sobre como o Waves está usando o Amazon Neptune	Consulte How Waves runs user queries and recommendations at scale with Amazon Neptune, por Pavel Vasilyev.	16 de junho de 2020
Versão 1.0.1.2 do mecanismo	Desde 10/06/2020, a versão 1.0.1.2 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.0.1.2.	10 de junho de 2020
Postagem no blog sobre como criar um repositório de conhecimentos do cliente	Consulte Building a customer 360 knowledge repository with Amazon Neptune and Amazon Redshift, por Ram Bhandarkar.	9 de junho de 2020
Postagem no blog sobre como a Gunosy está usando o Amazon Neptune	Consulte How Gunosy built a comment feature in News Pass using Amazon Neptune, por Yosuke Uchiyama.	8 de junho de 2020

Publicação no blog sobre o gráfico de conhecimento da AWS COVID-19	Consulte Construindo e consultando o gráfico de conhecimento da AWS COVID-19 de Ninad Kulkarni, Colby Wise, George Price e Miguel Romero.	2 de junho de 2020
Postagem no blog sobre como explorar a pesquisa de COVID-19 usando o Amazon Neptune	Consulte Como explorar pesquisas científicas sobre COVID-19 com o Amazon Neptune, o Amazon Comprehend Medical e o Tom Sawyer Graph Database Browser por George Price, Colby Wise, Miguel Romero e Ninad Kulkarni.	2 de junho de 2020
Agora você pode carregar dados em Neptune usando AWS DMS	Consulte Uso do AWS Database Migration Service para carregar dados no Amazon Neptune a partir de um armazenamento de dados diferente.	1 de junho de 2020
A versão 1.0.2.0 do mecanismo está obsoleta.	A versão 1.0.2.0 do mecanismo Amazon Neptune agora está obsoleta. Os clusters em execução nesta versão do mecanismo serão atualizados automatic amente para a versão 1.0.2.1 durante a primeira janela de manutenção após 1º de junho de 2020.	19 de maio de 2020

Postagem no blog sobre como criar um grafo de identidade do cliente usando o Neptune	Consulte Building a customer identity graph with Amazon Neptune por Rajesh Wunnava e Taylor Riggan.	12 de maio de 2020
Versão 1.0.2.0.R3 do mecanismo	Desde 05/05/2020, a versão 1.0.2.0.R3 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.0.2.0.R3.	5 de maio de 2020
Versão 1.0.2.1.R6 do mecanismo	Desde 22/04/2020, a versão 1.0.2.1.R6 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.0.2.1.R6.	22 de abril de 2020
Postagem no blog sobre migração de dados do Neo4j para o Neptune	Consulte Migrating a Neo4j graph database to Amazon Neptune with a fully automated utility por Sanjeet Sahay.	13 de abril de 2020

Postagem no blog sobre redução de custos da criação de aplicações de grafos com o Neptune	Consulte Lower the cost of building graph apps by up to 76% with Amazon Neptune T3 instances por Karthik Bharathy e Brad Bebee.	9 de abril de 2020
O Neptune oferece uma classe de instância intermitente T3.	Agora você pode criar uma instância de intermitência T3 do Amazon Neptune para fins de desenvolvimento e teste econômicos. Consulte a Classe de instância de intermitência T3 do Neptune.	8 de abril de 2020
Versão 1.0.2.2.R2 do mecanismo	Desde 02/04/2020, a versão 1.0.2.2.R2 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.0.2.2.R2.	2 de abril de 2020
Postagem no blog sobre representação gráfica de dependência de investimento em EDGAR	Consulte Graphing investment dependency with Amazon Neptune por Lawrence Verdi.	17 de março de 2020
Lançamento do Neptune na Europa (Paris)	O Amazon Neptune agora está disponível na Europa (Paris) (eu-west-3).	11 de março de 2020

Versão 1.0.2.2 do mecanismo	Desde 09/03/2020, a versão 1.0.2.2 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões. Para obter mais informações sobre essa versão do mecanismo, consulte Neptune Engine Release 1.0.2.2.	9 de março de 2020
Interromper e reiniciar um cluster de banco de dados	Agora você pode interromper um cluster de banco de dados por 7 dias usando o console do Neptune e, posteriormente, reiniciá-lo quando precisar dele novamente. Enquanto seu cluster de banco de dados estiver interrompido, você será cobrado somente pelo armazenamento do cluster, pelos snapshots manuais e pelo armazenamento do backup automático, mas não pelas horas da instância de banco de dados. Consulte Interromper e iniciar um cluster de banco de dados do Amazon Neptune.	19 de fevereiro de 2020

Vídeo sobre um grafo social na Nike	Ouça as AWS conversas de Todd Escalona com Marc Wangenheim, gerente sênior de engenharia da Nike, sobre como a empresa potencializa vários aplicativos por meio de um gráfico social criado no Amazon Neptune. Consulte Nike: A Social Graph at Scale with Amazon Neptune.	11 de fevereiro de 2020
Os clusters do Neptune agora podem ser configurados para exigir conexões SSL.	Nas regiões que ainda oferecem suporte a conexões HTTP, o SSL agora é ativado por padrão em todos os novos grupos de parâmetros. Não há alterações nos grupos de parâmetros existentes, mas você pode forçar os clientes a usar o SSL alterando o parâmetro `neptune_enforce_ssl` para 1. Consulte Encryption in Transit: Connecting to Neptune Using SSL/HTTPS para obter informações sobre como habilitar conexões HTTP para um cluster em uma região que ainda oferece suporte a elas. Consulte Parameters That You Can Use to Configure Amazon Neptune para obter uma descrição dos parâmetros de cluster e instância.	10 de fevereiro de 2020

Agora você pode especificar a versão do mecanismo e a proteção contra exclusão no modelo do Neptune CloudFormation	O Amazon Neptune atualizou CloudFormation seu modelo para incluir `AWS::Neptune::DBCluster.EngineVersion` um parâmetro que permite especificar uma versão específica do mecanismo para seu novo cluster de banco de dados e `AWS::Neptune::DBCluster.DeletionProtection` um parâmetro que permite ativar a proteção contra exclusão para ele.	9 de fevereiro de 2020
Deletion protection (Proteção contra exclusão)	O Amazon Neptune forneceu proteção contra exclusão para clusters de banco de dados e instâncias. Enquanto a proteção contra exclusão estiver habilitada em um cluster ou instância de banco de dados, não será possível excluí-la. Consulte Não é possível excluir uma instância de banco de dados se a proteção contra exclusão está habilitada.	20 de janeiro de 2020
Lançamento do Neptune na China (Ningxia)	O Amazon Neptune já está disponível na China (Ningxia) (cn-northwest-1).	15 de janeiro de 2020

Versão 1.0.2.1.R4 do mecanismo	O patch R4 para a versão 1.0.2.1 do mecanismo está disponível ao público em geral. Para obter mais informações, consulte Neptune Engine Release 1.0.2.1.R4.	20 de dezembro de 2019
Versão 1.0.2.1.R3 do mecanismo	O patch R3 para a versão 1.0.2.1 do mecanismo está disponível ao público em geral. Para obter mais informações, consulte Neptune Engine Release 1.0.2.1.R3.	12 de dezembro de 2019
Postagem no blog sobre como usar o Neptune para analisar feeds de redes sociais	Consulte Analisar feeds de mídia social usando o Amazon Neptune.	27 de novembro de 2019
Versão 1.0.2.1.R2 do mecanismo	O patch R2 para a versão 1.0.2.1 do mecanismo está disponível ao público em geral. Para obter mais informações, consulte Neptune Engine Release 1.0.2.1.R2.	25 de novembro de 2019
Versão 1.0.2.1.R1 do mecanismo	A versão 1.0.2.1.R1 do mecanismo Amazon Neptune está disponível ao público em geral. Para obter mais informações, consulte Neptune Engine Release 1.0.2.1.	22 de novembro de 2019

Versão 1.0.2.0.R2 do mecanismo	O patch R2 para a versão 1.0.2.0 do mecanismo está disponível ao público em geral. Para obter mais informações, consulte Neptune Engine Release 1.0.2.0.R2.	21 de novembro de 2019
Postagem no blog sobre sessões e workshops do Neptune na re:Invent 2019	Veja seu guia para sessões, workshops e palestras sobre o Amazon Neptune no re:Invent 2019. AWS	20 de novembro de 2019
Versão 1.0.2.0.R1 do mecanismo	A versão 1.0.2.0.R1 do mecanismo Amazon Neptune está disponível ao público em geral. Para obter mais informações, consulte Neptune Engine Release 1.0.2.0.	8 de novembro de 2019
Postagem no blog sobre a captura de alterações de grafos usando o Neptune Streams	Consulte Capturar alterações de gráficos usando Neptune Streams.	6 de novembro de 2019
Versão 1.0.1.0.200502.0 do mecanismo	A versão 1.0.1.0.200502.0 do mecanismo do Amazon Neptune está disponível ao público em geral. Para obter mais informações, consulte a Atualização 1.0.1.0.200502.0.	31 de outubro de 2019
Lançamento do Neptune no Oriente Médio (Bahrein)	O Amazon Neptune já está disponível no Oriente Médio (Bahrein) (me-south-1).	30 de outubro de 2019

Lançamento do Neptune no Canadá (Central)	O Amazon Neptune agora está disponível no Canadá (Central) (ca-central-1).	30 de outubro de 2019
Postagem no blog sobre o novo atributo SPARQL Streams do Neptune e suporte a consultas federadas SPARQL	Consulte Amazon Neptune releases Streams, SPARQL federated query for graphs and more.	17 de outubro de 2019
Versão 1.0.1.0.200463.0 do mecanismo	A versão 1.0.1.0.200463.0 do mecanismo do Amazon Neptune está disponível ao público em geral. Para obter mais informações, consulte a Atualização 1.0.1.0.200463.0.	15 de outubro de 2019
Versão 1.0.1.0.200457.0 do mecanismo	A versão 1.0.1.0.200457.0 do mecanismo do Amazon Neptune está disponível ao público em geral. Para obter mais informações, consulte a Atualização 1.0.1.0.200457.0.	19 de setembro de 2019
Postagem no blog sobre o novo atributo explain do SPARQL	Consulte Using SPARQL explain to understand query execution in Amazon Neptune.	17 de setembro de 2019
Postagem no blog sobre o suporte do Neptune para 3.4 TinkerPop	Veja: o Amazon Neptune agora TinkerPop oferece suporte a 3.4 recursos.	6 de setembro de 2019
Postagem no blog sobre como usar o Neptune PyTorch na Amazon SageMaker	Veja uma experiência personalizada “compre por estilo” usando na Amazon e no PyTorch Amazon SageMaker Neptune.	22 de agosto de 2019

Publicação no blog sobre o uso do Neptune com AWS AppSync o Amazon Elasticac he	Consulte Integração de fontes de dados alternativas com AWS AppSync: Amazon Neptune e Amazon. ElastiCac he	22 de agosto de 2019
Neptune foi lançado AWS GovCloud em (Leste dos EUA)	O Amazon Neptune agora está disponível AWS GovCloud em (Leste dos EUA) (us-gov-east-1).	21 de agosto de 2019
Neptune foi lançado AWS GovCloud em (Oeste dos EUA)	O Amazon Neptune agora está disponível AWS GovCloud em (US-West) (us-gov-west-1).	14 de agosto de 2019
Versão 1.0.1.0.200369.0 do mecanismo	A versão 1.0.1.0.200369.0 do mecanismo do Amazon Neptune está disponível ao público em geral. Para obter mais informações, consulte a Atualização 1.0.1.0.200369.0.	13 de agosto de 2019
Versão 1.0.1.0.200366.0 do mecanismo	A versão 1.0.1.0.200366.0 do mecanismo do Amazon Neptune está disponível ao público em geral. Para obter mais informações, consulte a Atualização 1.0.1.0.200366.0.	26 de julho de 2019
Postagem no blog sobre como usar o Neptune PyTorch na Amazon SageMaker	Veja uma experiência personalizada “compre por estilo” usando na Amazon e no PyTorch Amazon SageMaker Neptune.	3 de julho de 2019

Versão 1.0.1.0.200348.0 do mecanismo	A versão 1.0.1.0.200348.0 do mecanismo do Amazon Neptune está disponível ao público em geral. Para obter mais informações, consulte a Atualização 1.0.1.0.200348.0.	2 de julho de 2019
Lançamento do Neptune na Europa (Estocolmo)	O Amazon Neptune agora está disponível na Europa (Estocolmo) (eu-north-1).	27 de junho de 2019
O Neptune agora pode publicar registros de auditoria no Logs CloudWatch	Para obter mais informaçõ es, consulte Publicação do Neptune Logs no Amazon Logs. CloudWatch	18 de junho de 2019
Versão 1.0.1.0.200310.0 do mecanismo	A versão 1.0.1.0.200310.0 do mecanismo do Amazon Neptune está disponível ao público em geral. Para obter mais informações, consulte a Atualização 1.0.1.0.200310.0.	12 de junho de 2019
Postagem no blog LifeOmic sobre JupiterOne	Veja como JupiterOne simplific a LifeOmic as operações de segurança e conformidade com o Amazon Neptune.	2 de maio de 2019
Lançamento do Neptune na Ásia-Pacífico (Seul)	O Amazon Neptune já está disponível na Ásia-Pacífico (Seul) (ap-northeast-2).	1º de maio de 2019
Versão 1.0.1.0.200296.0 do mecanismo	A versão 1.0.1.0.200296.0 do mecanismo do Amazon Neptune está disponível ao público em geral. Para obter mais informações, consulte a Atualização 1.0.1.0.200296.0.	1º de maio de 2019

Lançamento do Neptune na Ásia-Pacífico (Mumbai)	O Amazon Neptune já está disponível na Ásia-Pacífico (Mumbai) (ap-south-1).	6 de março de 2019
Postagem no blog sobre dicas relacionadas a consultas do Gremlin	Consulte Apresentação de dicas sobre consultas do Gremlin para o Amazon Neptune.	26 de fevereiro de 2019
Lançamento do Neptune na Ásia-Pacífico (Tóquio)	O Amazon Neptune já está disponível na Ásia-Pacífico (Tóquio) (ap-northeast-1).	23 de janeiro de 2019
AWS CloudFormation modelo para criar uma AWS Lambda função para acessar Neptune	Atualizou a seção de introduçã o e adicionou um AWS CloudFormation modelo para criar uma função Lambda para usar com o Neptune. Para obter mais informações, consulte Conceitos básicos do Neptune.	23 de janeiro de 2019
Versão 1.0.1.0.200267.0 do mecanismo	A versão 1.0.1.0.200267.0 do mecanismo do Amazon Neptune está disponível ao público em geral. Para obter mais informações, consulte a Atualização 1.0.1.0.200267.0.	21 de janeiro de 2019
Lançamento do Neptune na Ásia-Pacífico (Sydney)	O Amazon Neptune já está disponível na Ásia-Pacífico (Sydney) (ap-southeast-2).	9 de janeiro de 2019
Postagem no blog sobre como usar o Metaphactory	Consulte Como explorar gráficos de conhecimento no Amazon Neptune com o uso do Metaphactory.	9 de janeiro de 2019

Lançamento do Neptune na Ásia-Pacífico (Singapura)	O Amazon Neptune já está disponível na Ásia-Pacífico (Singapura) (ap-southeast-1).	13 de dezembro de 2018
Versão 1.0.1.0.200264.0 do mecanismo	A versão 1.0.1.0.200264.0 do mecanismo do Amazon Neptune está disponível ao público em geral. Para obter mais informações, consulte a Atualização 1.0.1.0.200264.0.	19 de novembro de 2018
Suporte a SSL do Amazon Neptune	O Neptune agora aceita conexões SSL.	19 de novembro de 2018
Tópicos de erros consolidados	Toda a mensagem de erro e informação de código está agora em um único tópico.	15 de novembro de 2018
Tópico de conceitos básicos atualizado	Atualização do tópico de conceitos básicos com links adicionais e documentação reorganizada.	14 de novembro de 2018
Versão 1.0.1.0.200258.0 do mecanismo	A versão 1.0.1.0.200258.0 do mecanismo do Amazon Neptune está disponível ao público em geral. Para obter mais informações, consulte a Atualização 1.0.1.0.200258.0.	8 de novembro de 2018
Lançamento do Neptune na Europa (Frankfurt)	O Amazon Neptune já está disponível na Europa (Frankfurt) (eu-central-1).	7 de novembro de 2018
Postagem no blog nº 1 em uma série	Consulte Deixe-me criar um gráfico para você — Parte 1 — Rotas aéreas.	7 de novembro de 2018

Postagem no blog sobre o uso de notebooks Amazon SageMaker Jupyter	Consulte Analisar gráficos do Amazon Neptune usando os notebooks Amazon Jupyter. SageMaker	1 de novembro de 2018
Versão 1.0.1.0.200255.0 do mecanismo	A versão 1.0.1.0.200255.0 do mecanismo do Amazon Neptune está disponível ao público em geral. Para obter mais informações, consulte a Atualização 1.0.1.0.200255.0.	29 de outubro de 2018
Lançamento do Neptune na Europa (Londres)	O Amazon Neptune agora está disponível na Europa (Londres) (eu-west-2).	3 de outubro de 2018
Versão 1.0.1.0.200237.0 do mecanismo	A versão 1.0.1.0.200237.0 do mecanismo do Amazon Neptune está disponível ao público em geral. Para obter mais informações, consulte a Atualização 1.0.1.0.200237.0.	6 de setembro de 2018
Versão 1.0.1.0.200236.0 do mecanismo	A versão 1.0.1.0.200236.0 do mecanismo do Amazon Neptune está disponível ao público em geral. Para obter mais informações, consulte a Atualização 1.0.1.0.200236.0.	24 de julho de 2018
Versão 1.0.1.0.200233.0 do mecanismo	A versão 1.0.1.0.200233.0 do mecanismo do Amazon Neptune está disponível ao público em geral. Para obter mais informações, consulte a Atualização 1.0.1.0.200233.0.	22 de junho de 2018

Novo início rápido do Neptune	Início rápido atualizado com o AWS CloudFormation tutorial do Gremlin Console. Para obter mais informações, consulte Amazon Neptune Quick Start Using. AWS CloudFormation	19 de junho de 2018
Versão inicial do Amazon Neptune	Essa é a versão inicial do Guia do usuário do Neptune. Consulte também a publicação sobre versão no blog, Amazon Neptune geralmente disponível.	30 de maio de 2018
Postagem no blog introdutória sobre o Neptune	Consulte Amazon Neptune – serviço totalmente gerenciado de banco de dados gráfico.	29 de novembro de 2017

Conceitos básicos do Amazon Neptune

O Amazon Neptune é um serviço de banco de dados de grafos totalmente gerenciado que pode ser escalado para lidar com bilhões de relacionamentos e permite consultá-los com latência de milissegundos, por um custo baixo para esse tipo de capacidade.

Se você estiver procurando informações mais detalhadas sobre o Neptune, consulte Visão geral dos atributos do Amazon Neptune.

Se você já conhece grafos, acesse Usar blocos de anotações de grafos. Ou se quiser criar um banco de dados do Neptune imediatamente, consulte Usando uma AWS CloudFormation pilha para criar um cluster de banco de dados Neptune.

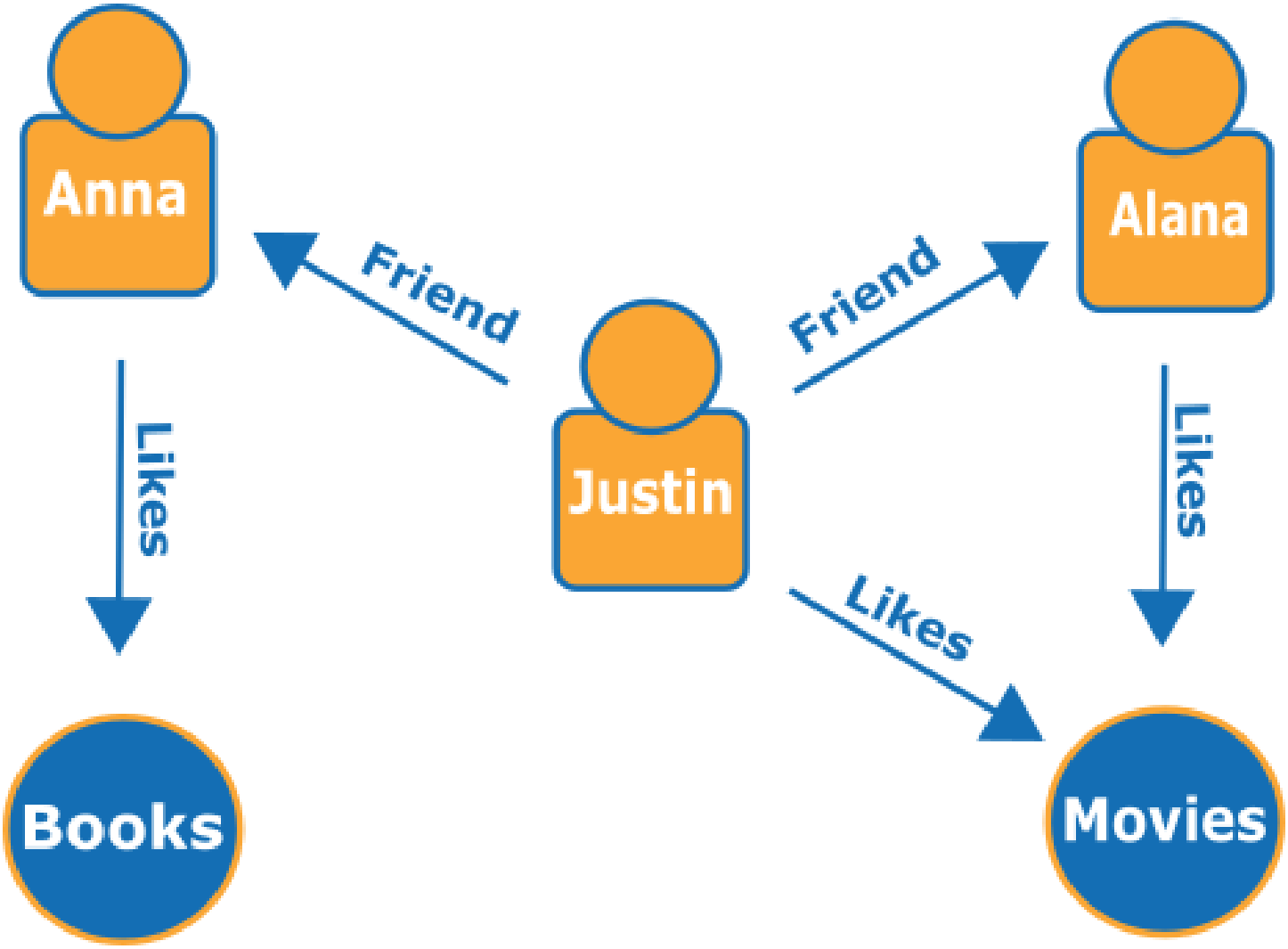
Caso contrário, é recomendável saber um pouco mais sobre bancos de dados de grafos antes de começar.

Do que exatamente se trata um banco de dados de grafos?

Os bancos de dados de grafos são otimizados para armazenar e consultar os relacionamentos entre os itens de dados.

Eles armazenam itens de dados como vértices do grafo e os relacionamentos entre eles como bordas. Cada borda tem um tipo e é direcionada de um vértice (o início) para outro (o final). Os relacionamentos podem ser chamados de predicados e também de bordas, e os vértices às vezes também são chamados de nós. Nos chamados grafos de propriedades, tanto os vértices quanto as bordas também podem ter propriedades adicionais associadas a eles.

Veja um pequeno grafo representando amigos e hobbies em uma rede social:



As bordas são mostradas como setas nomeadas e os vértices representam pessoas e hobbies específicos com os quais elas se conectam.

Um simples percurso desse gráfico pode dizer o que os amigos de Justin curtem.

Por que usar um banco de dados de grafos?

Um banco de dados de grafos será uma escolha natural sempre que conexões ou relacionamentos entre entidades forem um elemento central dos dados que você está tentando modelar.

Por um lado, é fácil modelar as interconexões de dados como um grafo e, depois, redigir consultas complexas que extraiam informações reais do grafo.

A criação de uma aplicação equivalente usando um banco de dados relacional exige que você crie várias tabelas com várias chaves externas e, depois, redija consultas SQL aninhadas e junções

complexas. Essa abordagem não apenas se torna rapidamente complicada do ponto de vista da codificação, mas o desempenho se degrada rapidamente à medida que a quantidade de dados aumenta.

Por outro lado, um banco de dados de grafos como o Neptune pode consultar relacionamentos entre bilhões de vértices sem travar.

O que você pode fazer com um banco de dados de grafos?

Os grafos podem representar as inter-relações de entidades do mundo real de várias maneiras, em termos de ações, propriedade, parentesco, opções de compra, conexões pessoais, laços familiares, etc.

Veja algumas das áreas mais comuns em que bancos de dados de grafos são usados:

- Grafos de conhecimento: os grafos de conhecimento permitem organizar e consultar todos os tipos de informações conectadas para responder a perguntas gerais. Usando um grafo de conhecimento, é possível adicionar informações tópicas aos catálogos de produtos e modelar diversas informações, como as contidas no Wikidata.

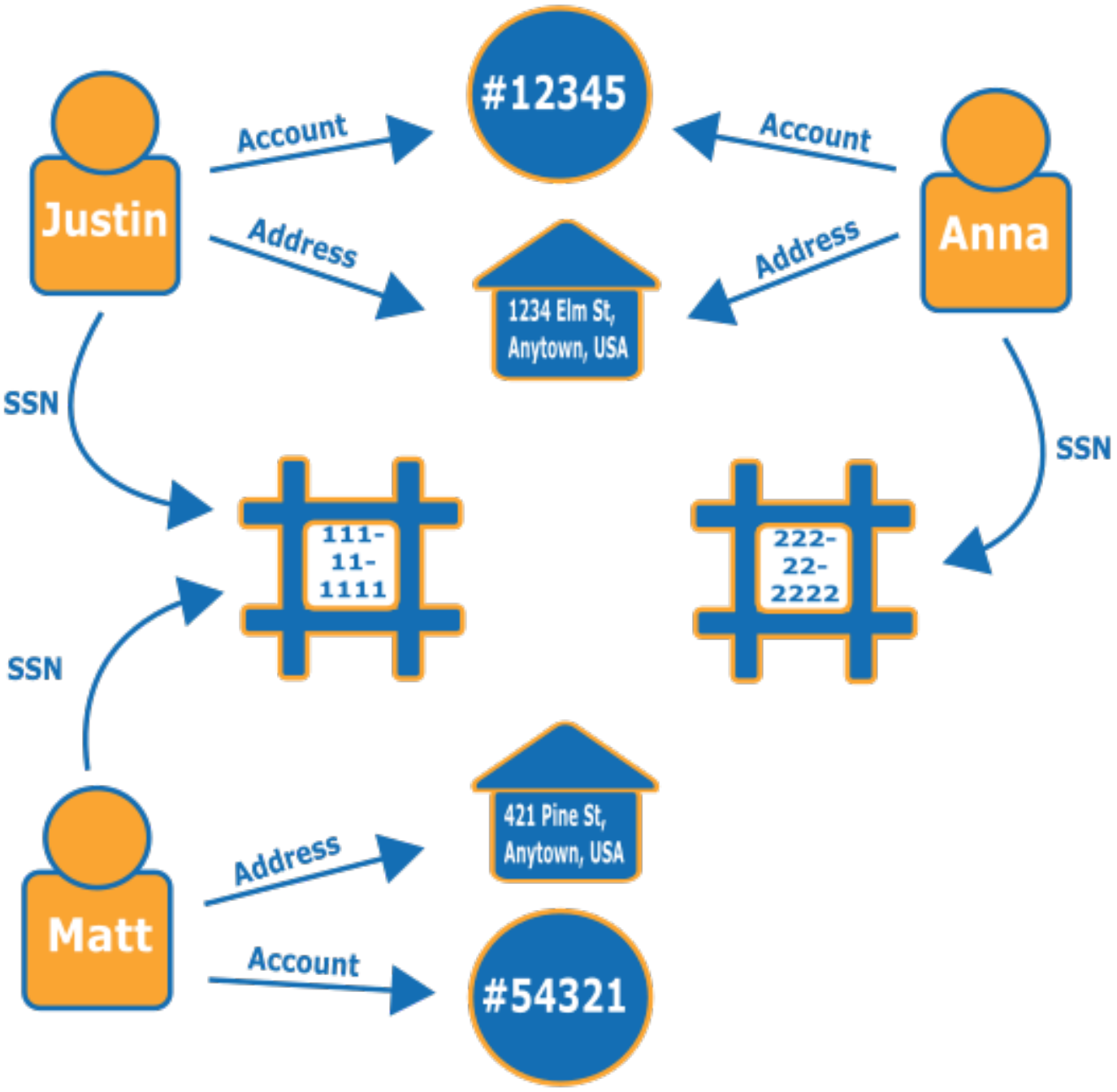
  Para saber mais sobre como os grafos de conhecimento funcionam e onde estão sendo usados, consulte Knowledge Graphs on AWS.
- Grafos de identidade: em um banco de dados de grafos, é possível armazenar relacionamentos entre categorias de informações, como interesses de clientes, amigos e histórico de compras, e depois consultar esses dados para fazer recomendações personalizadas e relevantes.

  Por exemplo, é possível usar um banco de dados de grafos para fazer recomendações de produtos a um usuário com base em produtos que são comprados por outros que são adeptos do mesmo esporte e têm histórico de compras semelhante. Ou você pode identificar pessoas que têm um amigo em comum, mas que ainda não se conhecem e fazer uma recomendação de amizade.

  Grafos desse tipo são conhecidos como grafos de identidade e são amplamente usados para personalizar as interações com os usuários. Para saber mais, consulte Identity Graphs on AWS. Para começar a criar o próprio grafo de identidade, é possível começar com o exemplo Identity Graph Using Amazon Neptune.
- Grafos de fraude: é um uso comum de bancos de dados de grafos. Eles podem ajudar você a monitorar compras com cartão de crédito e locais de compra para detectar um uso não característico ou detectar que um comprador está tentando usar o mesmo endereço de e-mail e cartão de crédito usados em um caso de fraude conhecido. Eles podem permitir que você confira

se há várias pessoas associadas a um endereço de e-mail pessoal ou várias pessoas em locais físicos diferentes que compartilham o mesmo endereço IP.

Examine o grafo a seguir. O grafo a seguir mostra o relacionamento de três pessoas e respectivas informações de identidade. Cada pessoa tem um endereço, uma conta bancária e um número de identidade. No entanto, é possível ver que Matt e Justin compartilham o mesmo número de previdência social, o que é irregular e indica uma possível fraude realizada por um dos dois. Uma consulta a um grafo de fraude pode revelar conexões desse tipo para que possam ser avaliadas.



Para saber mais sobre grafos de fraude e onde estão sendo usados, consulte Fraud Graphs on AWS.

- Redes sociais: uma das primeiras áreas e mais comuns em que bancos de dados de grafos são usados é em aplicações de redes sociais.

Por exemplo, suponha que você queira criar um feed social em um site. É possível usar com facilidade um banco de dados de grafos no back-end para fornecer resultados aos usuários que reflitam as atualizações mais recentes das respectivas famílias, amigos, pessoas cujas atualizações eles “curtem” e pessoas que moram perto deles.

- Instruções de direção: um grafo pode ajudar a encontrar a melhor rota de um ponto de partida até um destino, considerando-se o tráfego atual e os padrões de tráfego típicos.
- Logística: os grafos podem ajudar a identificar a maneira mais eficiente de usar os recursos de remessa e distribuição disponíveis para atender às necessidades do cliente.
- Diagnóstico: os grafos podem representar árvores de diagnóstico complexas que podem ser consultadas para identificar a origem das falhas e dos problemas observados.
- Pesquisa científica: com um banco de dados de grafos, é possível criar aplicações que armazenam e percorrem dados científicos e até mesmo informações médicas confidenciais usando criptografia em repouso. Por exemplo, é possível armazenar modelos de doença e interações genéticas. Você pode pesquisar padrões de gráficos em percursos proteicos para encontrar outros genes que podem ser associados a uma doença. É possível modelar compostos químicos como um grafo e consultar padrões em estruturas moleculares. Você pode correlacionar dados de pacientes de registros médicos em diferentes sistemas. Você pode organizar publicações de pesquisa por tópicos para encontrar informações relevantes rapidamente.
- Regras regulatórias: é possível armazenar requisitos regulatórios complexos na forma de grafos e consultá-los para detectar situações em que eles possam se aplicar às suas operações empresariais diárias.
- Topologia e eventos de rede: um banco de dados de grafos pode ajudar a gerenciar e proteger uma rede de TI. Ao armazenar a topologia da rede como um grafo, você também pode armazenar e processar vários tipos diferentes de eventos na rede. É possível responder a perguntas como quantos hosts estão executando uma aplicação específica. Você pode consultar padrões que possam mostrar que um host específico foi comprometido por um programa mal-intencionado e consultar dados de conexão que possam ajudar a rastrear o programa até o host original que o baixou.

Como consultar um grafo?

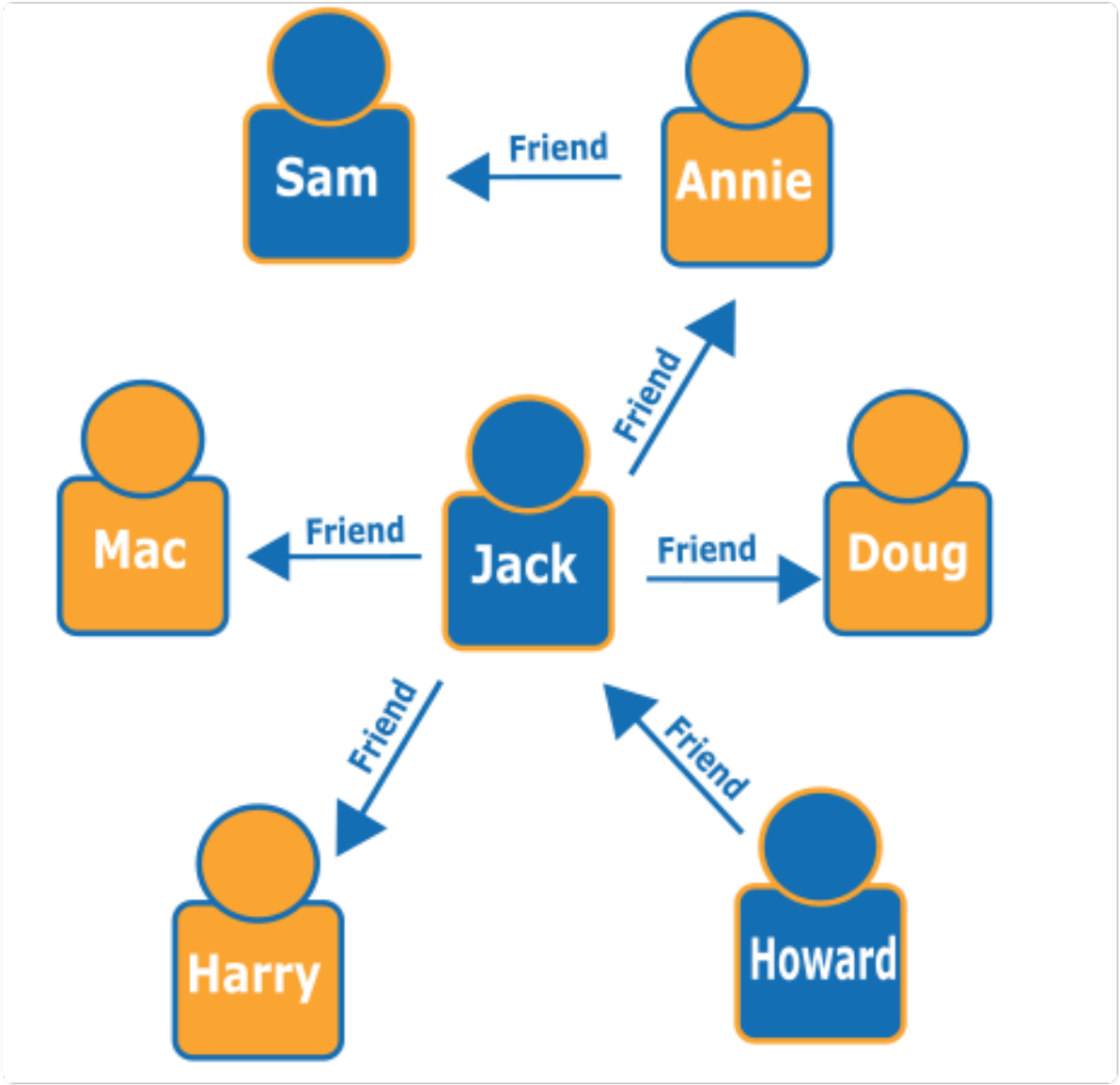
O Neptune é compatível com três linguagens de consulta para fins especiais projetadas para consultar dados de grafos de diferentes tipos. É possível usar essas linguagens para adicionar, modificar, excluir e consultar dados em um banco de dados de grafos do Neptune:

- O Gremlin é uma linguagem de percurso de grafos de propriedades. Uma consulta no Gremlin é um percurso composto por etapas distintas, cada uma das quais segue uma borda até um nó. Consulte a documentação do Gremlin em Apache TinkerPop3 para obter mais informações.

  A implementação do Gremlin no Neptune tem algumas diferenças em relação a outras implementações, principalmente ao usar o Gremlin-Groovy (as consultas do Gremlin enviadas como texto serializado). Para obter mais informações, consulte Conformidade com os padrões do Gremlin no Amazon Neptune.
- openCypher: openCypher é uma linguagem de consulta declarativa para grafos de propriedades originalmente desenvolvida pela Neo4j, que se tornou de código aberto em 2015, e contribuiu para o projeto openCypher sob uma licença de código aberto Apache 2. Consulte o documento Cypher Query Language Reference (Version 9) para saber a especificação da linguagem, bem como o Guia de Estilo da Cypher para obter informações adicionais.
- O SPARQL é uma linguagem de consulta declarativa para dados do RDF baseada na correspondência do padrão de grafo padronizado pelo World Wide Web Consortium (W3C) e descrito em SPARQL 1.1 Overview e na especificação SPARQL 1.1 Query Language. Consulte Conformidade com os padrões SPARQL no Amazon Neptune para obter detalhes específicos sobre a implementação do SPARQL no Neptune.

Exemplos de consultas correspondentes do Gremlin e do SPARQL

Considerando-se o seguinte grafo de pessoas (nós) e os respectivos relacionamentos (bordas), é possível descobrir quem são os "amigos dos amigos" de uma pessoa específica, por exemplo, os amigos dos amigos de Howard.



Analisando o gráfico, você pode ver que Howard tem um amigo, Jack, e que Jack tem quatro amigos: Annie, Harry, Doug e Mac. Esse é um exemplo simples com um gráfico simples, mas esses tipos de consultas podem escalar em complexidade, tamanho do conjunto de dados e tamanho do resultado.

Veja uma consulta de percurso do Gremlin que exibe os nomes dos amigos dos amigos de Howard:

```
g.V().has('name', 'Howard').out('friend').out('friend').values('name')
```

Veja uma consulta do SPARQL que exibe os nomes dos amigos dos amigos de Howard:

```
prefix : <#>

  select ?names where {
    ?howard :name "Howard" .
    ?howard :friend/:friend/:name ?names .
  }
```

> **Note**
>
> Cada parte de qualquer triplo de Resource Description Framework (RDF) tem uma URI associada. No exemplo acima, o prefixo do URI é intencionalmente curto.

Realizar um curso on-line sobre o uso do Amazon Neptune

Se você gosta de aprender com vídeos, a AWS oferece cursos on-line no AWS Online Tech Talks para ajudar você a começar. Um vídeo que apresenta bancos de dados de grafos é o seguinte:

Graph Database introduction, deep-dive and demo with Amazon Neptune.

Aprofundar-se na arquitetura de referência

Ao pensar nos problemas que um banco de dados de grafos pode resolver e como abordá-los, um dos melhores lugares para começar é o projeto do GitHub sobre arquiteturas de referência de grafos do Neptune.

Lá você pode encontrar descrições detalhadas dos tipos de workload de grafos e três seções que ajudam a criar um banco de dados de grafos eficaz:

- Data Models and Query Languages: essa seção mostra as diferenças entre o Gremlin e o SPARQL e como escolher entre eles.
- Graph Data Modeling: trata-se de uma discussão completa sobre como tomar decisões de modelagem de dados de grafos, incluindo orientações detalhadas sobre modelagem de grafos de propriedades usando Gremlin e modelagem do RDF usando SPARQL.
- Converting Other Data Models to a Graph Model: aqui você pode descobrir como converter um modelo de dados relacional em um modelo de grafos.

Há também três seções que orientam você nas etapas específicas para usar o Neptune:

- Connecting to Amazon Neptune from Clients Outside the Neptune VPC: essa seção mostra várias opções para se conectar ao Neptune de fora da VPC em que o cluster de banco de dados está localizado.
- Accessing Amazon Neptune from AWS Lambda Functions: aqui você descobrirá como se conectar de forma confiável ao Neptune pelas funções do Lambda.
- Writing to Amazon Neptune from an Amazon Kinesis Data Stream: essa seção pode ajudar você a lidar com cenários de alto throughput de gravação com o Neptune.

Usar os blocos de anotações de grafos Neptune para começar rapidamente

Não é necessário usar os blocos de anotações de grafos Neptune para trabalhar com um grafo do Neptune, então, se quiser, crie um banco de dados do Neptune imediatamente usando um modelo do AWS CloudFormation.

Ao mesmo tempo, se você não tem conhecimento sobre grafos e quer aprender e experimentar, ou se tem experiência e quer refinar suas consultas, a bancada de trabalho do Neptune oferece um ambiente de desenvolvimento interativo (IDE) que pode aumentar sua produtividade ao criar aplicações de grafos.

O Neptune fornece Jupyter e JupyterLabnotebooks no projeto de caderno gráfico Neptune de código aberto e na bancada de trabalho Neptune. GitHub Esses blocos de anotações oferecem exemplos de tutoriais de aplicações e trechos de código em um ambiente de codificação interativo onde você pode saber mais sobre a tecnologia de grafos e o Neptune. É possível usá-los para percorrer a instalação, a configuração, o preenchimento e a consulta de grafos usando diferentes linguagens de consulta, conjuntos de dados distintos e até mesmo bancos de dados diferentes no back-end.

É possível hospedar esses blocos de anotações de várias maneiras diferentes:

- A bancada de trabalho Neptune permite que você execute notebooks Jupyter em um ambiente totalmente gerenciado, hospedado na SageMaker Amazon, e carregue automaticamente a versão mais recente do projeto de caderno gráfico Neptune para você. É fácil configurar a bancada de trabalho no console do Neptune ao criar um banco de dados do Neptune.

> **Note**
>
> Ao criar uma instância do notebook Neptune, você tem duas opções de acesso à rede: acesso direto pela SageMaker Amazon (o padrão) e acesso por meio de uma VPC. Em qualquer uma das opções, o notebook requer acesso à Internet para buscar dependências de pacotes para instalar o Neptune Workbench. A falta de acesso à Internet fará com que a criação de uma instância do notebook Neptune falhe.

- Você também pode instalar o Jupyter localmente. Isso permite que você execute os blocos de anotações no laptop, conectado ao Neptune ou em uma instância local de um dos bancos de dados de grafos de código aberto. No último caso, é possível experimentar a tecnologia de grafos o quanto quiser sem gastar. Então, quando estiver com tudo pronto, você poderá migrar sem problemas para o ambiente de produção gerenciado oferecido pelo Neptune.

Usar a bancada de trabalho do Neptune para hospedar blocos de anotações Neptune

O Neptune oferece os tipos de instância T3 e T4g com os quais é possível começar por menos de USD 0,10 por hora. Você é cobrado pelos recursos do Workbench por meio da Amazon SageMaker, separadamente do faturamento do Neptune. Consulte a página Preços do Neptune. O Jupyter e JupyterLab os notebooks criados na bancada de trabalho Neptune usam um ambiente Amazon Linux 2 e 3. JupyterLab Para obter mais informações sobre o suporte a JupyterLab notebooks, consulte a SageMakerdocumentação da Amazon.

Você pode criar um Jupyter ou um JupyterLab notebook usando a bancada de trabalho Neptune de duas maneiras: AWS Management Console

- Use o menu Configuração de cadernos ao criar um cluster de banco de dados do Neptune. Para isso, siga as etapas descritas em Iniciar um cluster de banco de dados do Neptune usando o AWS Management Console.
- Use o menu Blocos de anotações no painel de navegação esquerdo depois que o cluster de banco de dados já tiver sido criado. Para isso, siga as etapas abaixo.

Para criar um Jupyter ou JupyterLab notebook usando o menu Notebooks

1. Faça login no AWS Management Console e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. No painel de navegação à esquerda, escolha Notebooks (Blocos de anotações).
3. Escolha Criar caderno.
4. Na lista Cluster, escolha o cluster de banco de dados do Neptune. Se você ainda não tiver um cluster de banco de dados, escolha Create cluster (Criar cluster) para criar um.
5. Selecione um Tipo de instância de caderno.
6. Insira um nome para o seu bloco de anotações e, opcionalmente, uma descrição.
7. A menos que você já tenha criado uma função AWS Identity and Access Management (IAM) para seus notebooks, escolha Criar uma função do IAM e insira um nome de função do IAM.

   > **Note**
   >
   > Se você optar por reutilizar um perfil do IAM criado para um bloco de anotações anterior, a política do perfil deverá conter as permissões corretas para acessar o cluster de banco de dados do Neptune que você está usando. É possível verificar isso conferindo se os componentes do ARN do recurso sob a ação `neptune-db:*` correspondem a esse cluster. Permissões configuradas incorretamente geram erros de conexão quando você tenta executar comandos mágicos do caderno.

8. Escolha Criar caderno. O processo de criação pode levar de cinco a dez minutos até que tudo esteja pronto.
9. Depois que seu notebook for criado, selecione-o e escolha Abrir Jupyter ou Abrir. JupyterLab

O console pode criar um perfil do AWS Identity and Access Management (IAM) para os blocos de anotações ou você mesmo pode criar um. A política para essa função deve incluir o seguinte:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
```

```
      ],
      "Resource": [
        "arn:aws:s3:::aws-neptune-notebook-(AWS region)",
        "arn:aws:s3:::aws-neptune-notebook-(AWS region)/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "neptune-db:*",
      "Resource": [
        "arn:aws:neptune-db:(AWS region):(AWS account ID):(Neptune resource ID)/*"
      ]
    }
  ]
}
```

Observe que a segunda declaração na política acima lista um ou mais IDs de recursos do cluster do Neptune.

Além disso, a função deve estabelecer a seguinte relação de confiança:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sagemaker.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Novamente, preparar tudo pode levar de cinco a dez minutos.

Você pode configurar o novo bloco de anotações para funcionar com o Neptune ML, conforme explicado em Configurar manualmente um bloco de anotações Neptune para o Neptune ML.

Usando Python para conectar um SageMaker notebook genérico ao Neptune

Conectar um notebook ao Netuno é fácil se você tiver instalado o Neptune Magics, mas também é possível conectar um notebook SageMaker ao Netuno usando Python, mesmo se você não estiver usando um notebook Neptune.

Etapas a serem seguidas para se conectar ao Neptune em uma célula de notebook SageMaker

1. Instale o cliente do Python em Gremlin:

   ```
   !pip install gremlinpython
   ```

   Os notebooks Neptune instalam o cliente Gremlin Python para você, então essa etapa só é necessária se você estiver usando um notebook simples. SageMaker
2. Escreva um código como o seguinte para se conectar e emitir uma consulta do Gremlin:

   ```
   from gremlin_python import statics
   from gremlin_python.structure.graph import Graph
   from gremlin_python.process.graph_traversal import __
   from gremlin_python.process.strategies import *
   from gremlin_python.driver.driver_remote_connection import DriverRemoteConnection
   from gremlin_python.driver.aiohttp.transport import AiohttpTransport
   from gremlin_python.process.traversal import *
   import os

   port = 8182
   server = '(your server endpoint)'

   endpoint = f'wss://{server}:{port}/gremlin'

   graph=Graph()

   connection = DriverRemoteConnection(endpoint,'g',

    transport_factory=lambda:AiohttpTransport(call_from_event_loop=True))

   g = graph.traversal().withRemote(connection)

   results = (g.V().hasLabel('airport')
                   .sample(10)
                   .order()
                   .by('code')
   ```

```
                .local(__.values('code','city').fold())
                .toList())

# Print the results in a tabular form with a row index
for i,c in enumerate(results,1):
    print("%3d %4s %s" % (i,c[0],c[1]))

connection.close()
```

> **Note**
>
> Se você estiver usando uma versão do cliente do Python em Gremlin anterior à 3.5.0, esta linha:
>
> ```
> connection = DriverRemoteConnection(endpoint,'g',
>
>  transport_factory=lambda:AiohttpTransport(call_from_event_loop=True))
> ```
>
> Será apenas:
>
> ```
> connection = DriverRemoteConnection(endpoint,'g')
> ```

Ativando CloudWatch registros nos notebooks Neptune

CloudWatch os registros agora estão habilitados por padrão para o Neptune Notebooks. Se você tiver um notebook antigo que não esteja produzindo CloudWatch registros, siga estas etapas para habilitá-los manualmente:

1. Faça login no AWS Management Console e abra o SageMaker console.
2. No painel de navegação à esquerda, selecione Bloco de anotações e depois Instâncias do bloco de anotações. Procure o nome do bloco de anotações Neptune para o qual você gostaria de habilitar os logs.
3. Acesse a página de detalhes selecionando o nome dessa instância do bloco de anotações.
4. Se a instância do bloco de anotações estiver em execução, selecione o botão Interromper, no canto superior direito da página de detalhes do bloco de anotações.

5. Em Permissões e criptografia, há um campo para o ARN do perfil do IAM. Selecione o link nesse campo para acessar o perfil do IAM com o qual essa instância de bloco de anotações é executada.
6. Crie a seguinte política:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogDelivery",
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:DeleteLogDelivery",
        "logs:Describe*",
        "logs:GetLogDelivery",
        "logs:GetLogEvents",
        "logs:ListLogDeliveries",
        "logs:PutLogEvents",
        "logs:PutResourcePolicy",
        "logs:UpdateLogDelivery"
      ],
      "Resource": "*"
    }
  ]
}
```

7. Salve essa nova política e anexe-a ao perfil do IAM encontrado na etapa 4.
8. Clique em Iniciar no canto superior direito da página de detalhes da instância do SageMaker notebook.
9. Quando os logs começarem a fluir, você verá um link Visualizar logs abaixo do campo Configuração do ciclo de vida, na parte inferior esquerda da seção Configurações da instância do bloco de anotações na página de detalhes.

Se um notebook falhar ao iniciar, haverá uma mensagem na página de detalhes do notebook no SageMaker console, informando que a instância do notebook demorou mais de 5 minutos para iniciar. CloudWatch registros relevantes para esse problema podem ser encontrados com este nome:

```
(your-notebook-name)/LifecycleConfigOnStart
```

Configurar blocos de anotações de grafos na máquina local

O projeto de bloco de anotações de grafos tem instruções para configurar os blocos de anotações Neptune na máquina local:

- Pré-requisitos
- Jupyter e instalação JupyterLab
- Conectar-se a um banco de dados de grafos

É possível conectar os blocos de anotações locais a um cluster de banco de dados do Neptune ou a uma instância local ou remota de um banco de dados de grafos de código aberto.

Usar blocos de anotações Neptune com clusters do Neptune

Se você estiver se conectando a um cluster Neptune no back-end, talvez queira executar os notebooks na Amazon. SageMaker Conectar-se ao Neptune a SageMaker partir de pode ser mais conveniente do que a partir de uma instalação local dos notebooks e permitirá que você trabalhe mais facilmente com o Neptune ML.

Para obter instruções sobre como configurar cadernos em SageMaker, consulte Lançamento de cadernos gráficos usando a Amazon. SageMaker

Para obter instruções sobre como instalar e configurar o Neptune em si, consulte Configurar o Neptune.

Você também pode conectar uma instalação local dos blocos de anotações Neptune a um cluster de banco de dados do Neptune. Isso pode ser um pouco mais complicado porque os clusters de banco de dados do Amazon Neptune só podem ser criados em uma Amazon Virtual Private Cloud (VPC), que é, por design, isolada do mundo externo. Há várias maneiras de se conectar a uma VPC de fora dela. Uma delas é usar um balanceador de carga. Outra é usar o emparelhamento de VPCs (consulte o Guia de emparelhamento da Amazon Virtual Private Cloud).

No entanto, a maneira mais conveniente para a maioria das pessoas é se conectar para configurar um servidor proxy do Amazon EC2 na VPC e, depois, usar o tunelamento SSH (também chamado de encaminhamento de porta) para se conectar a ele. Você pode encontrar instruções sobre

como configurar em Conectando o caderno gráfico localmente ao Amazon Neptune na pasta `additional-databases/neptune` do projeto do notebook gráfico. GitHub

Usar blocos de anotações Neptune com bancos de dados de grafos de código aberto

Para começar a usar a tecnologia de grafos sem nenhum custo, você também pode usar os blocos de anotações Neptune com vários bancos de dados de código aberto no back-end. Exemplos são o servidor TinkerPop Gremlin e o banco de dados Blazegraph.

Para usar o servidor do Gremlin como o banco de dados de back-end, siga as instruções em:

- O caderno gráfico Conectando a uma pasta do Gremlin Server. GitHub
- A pasta de configuração do Gremlin do notebook gráfico. GitHub

Para usar uma instância local do Blazegraph como o banco de dados de back-end, siga estas instruções:

- Instruções de início rápido do Blazegraph
- A pasta de configuração Blazegraph do notebook gráfico. GitHub

Migrando seus cadernos Neptune do Jupyter para o 3 JupyterLab

Os blocos de anotações Neptune criados antes de 21 de dezembro de 2022 usam o ambiente Amazon Linux 1. Você pode migrar notebooks Jupyter antigos criados antes dessa data para o novo ambiente Amazon Linux 2 com JupyterLab 3 seguindo as etapas descritas nesta postagem do AWS blog: Migre seu trabalho para uma SageMaker instância de notebook Amazon com o Amazon Linux 2.

Além disso, também há mais algumas etapas que se aplicam especificamente à migração dos blocos de anotações Neptune para o novo ambiente:

Pré-requisitos específicos do Neptune

No perfil do IAM do bloco de anotações Neptune de origem, adicione todas as seguintes permissões:

```
{
  "Effect": "Allow",
  "Action": [
    "s3:GetObject",
```

```
    "s3:ListBucket",
    "s3:CreateBucket",
    "s3:PutObject"
  ],
  "Resource": [
    "arn:aws:s3:::(your ebs backup bucket name)",
    "arn:aws:s3:::(your ebs backup bucket name)/*"
  ]
},
{
  "Effect": "Allow",
  "Action": [
    "sagemaker:ListTags"
  ],
  "Resource": [
    "*"
  ]
}
```

Especifique o ARN correto para o bucket do S3 que você usará para fazer backup.

Configuração de ciclo de vida específica do Neptune

Ao criar o segundo script de configuração do ciclo de vida para restaurar o backup (de `on-create.sh`) conforme descrito na postagem no blog, o nome do ciclo de vida deve seguir o formato `aws-neptune-*`, como `aws-neptune-sync-from-s3`. Isso garante que a LCC possa ser selecionada durante a criação do bloco de anotações no console do Neptune.

Sincronização específica do Neptune de um snapshot para uma nova instância

Nas etapas descritas na postagem no blog para sincronização de um snapshot para uma nova instância, veja as alterações específicas do Neptune:

- Na etapa 4, escolha notebook-al2-v2.
- Na etapa 5, reutilize o perfil do IAM do bloco de anotações Neptune de origem.
- Entre as etapas 7 e 8:
  - Em Configurações da instância do bloco de anotações, defina um nome que use o formato `aws-neptune-*`.
  - Abra o menu sanfonado de configurações de Rede e selecione a mesma VPC, sub-rede e grupo de segurança do bloco de anotações de origem.

Etapas específicas do Neptune após a criação do bloco de anotações

1. Selecione o botão Abrir Jupyter para o bloco de anotações. Quando o arquivo `SYNC_COMPLETE` aparecer no diretório principal, vá para a próxima etapa.
2. Acesse a página da instância do notebook no SageMaker console.
3. Interrompa o bloco de anotações.
4. Selecione Edit (Editar).
5. Nas configurações da instância do bloco de anotações, edite o campo Configuração do ciclo de vida selecionando o ciclo de vida original do bloco de anotações Neptune de origem. Observe que esse não é o ciclo de vida de backup do EBS.
6. Selecione Atualizar configurações do bloco de anotações.
7. Inicie o bloco de anotações novamente.

Com as modificações descritas aqui nas etapas descritas na postagem do blog, seus cadernos gráficos agora devem ser migrados para uma nova instância do notebook Neptune que usa o ambiente Amazon Linux 2 e 3. JupyterLab Eles aparecerão para acesso e gerenciamento na página Netuno no, e agora você pode continuar seu trabalho de onde parou selecionando Abrir Jupyter ou Abrir. AWS Management Console JupyterLab

Usar as magias da bancada de trabalho do Neptune nos caderno

A bancada de trabalho do Neptune oferece vários comandos chamados magias nos cadernos que economizam muito tempo e esforço. Eles se enquadram em duas categorias: magias de linha e magias de célula.

Magias de linha são comandos precedidos por um único sinal de porcentagem (%). Eles recebem apenas entrada de linha, não entrada do resto do corpo da célula. A bancada de trabalho do Neptune oferece as seguintes magias de linha:

- %seed
- %load
- %load_ids
- %load_status
- %cancel_load
- %status

- %gremlin_status
- %opencypher_status ou %oc_status
- %stream_viewer
- %sparql_status
- %graph_notebook_config
- %graph_notebook_host
- %graph_notebook_version
- %graph_notebook_vis_options
- %statistics
- %summary

As magias de célula são precedidas por dois sinais de porcentagem (%%) em vez de um e usa o conteúdo da célula como entrada, embora também possam usar o conteúdo da linha como entrada. A bancada de trabalho do Neptune oferece as seguintes magias de célula:

- %%sparql
- %%gremlin
- %%opencypher ou %%oc
- %%graph_notebook_config
- %%graph_notebook_vis_options

Há também duas magias, de linha e de célula, para trabalhar com Machine learning no Neptune:

- %neptune_ml
- %%neptune_ml

> **Note**
>
> Ao trabalhar com magias do Neptune, geralmente é possível obter um texto de ajuda usando um parâmetro `--help` ou `-h`. Com uma magia de célula, o corpo não pode ficar vazio; portanto, ao obter ajuda, coloque um texto de preenchimento, mesmo que seja um único caractere, no corpo. Por exemplo: .
>
> ```
> %%gremlin --help
> ```

```
x
```

Injeção de variável em magia de célula ou de linha

As variáveis definidas em um caderno podem ser referenciadas dentro de qualquer magia de célula ou linha no caderno usando o formato: `${VAR_NAME}`.

Por exemplo, suponha que você defina estas variáveis:

```
c = 'code'
my_edge_labels = '{"route":"dist"}'
```

Depois, esta consulta do Gremlin em uma magia de célula:

```
%%gremlin -de $my_edge_labels
g.V().has('${c}','SAF').out('route').values('${c}')
```

É equivalente ao seguinte:

```
%%gremlin -de {"route":"dist"}
g.V().has('code','SAF').out('route').values('code')
```

Argumentos de consulta que funcionam com todas as linguagens de consulta

Os seguintes argumentos de consulta funcionam com as magias `%%gremlin`, `%%opencypher` e `%%sparql` na bancada de trabalho do Neptune:

Argumentos de consulta comuns

- **`--store-to`** (ou **`-s`**): especifica o nome de uma variável na qual armazenar os resultados da consulta.
- **`--silent`**: se estiver presente, nenhuma saída será exibida após a conclusão da consulta.
- **`--group-by`** (ou **`-g`**): especifica a propriedade usada para agrupar nós (como `code` ou `T.region`). Os vértices são coloridos com base no grupo atribuído.
- **`--ignore-groups`**: se estiverem presentes, todas as opções de agrupamento serão ignoradas.

- **`--display-property`** (ou **`-d`**): especifica a propriedade cujo valor deve ser exibido para cada vértice.

  O valor padrão para cada linguagem de consulta é o seguinte:

  - Para o Gremlin: `T.label`.
  - Para o openCypher: `~labels`.
  - Para o SPARQL: `type`.
- **`--edge-display-property`** (ou **`-t`**): especifica a propriedade cujo valor deve ser exibido para cada borda.

  O valor padrão para cada linguagem de consulta é o seguinte:

  - Para o Gremlin: `T.label`.
  - Para o openCypher: `~labels`.
  - Para o SPARQL: `type`.
- **`--tooltip-property`** (ou **`-de`**): especifica a propriedade cujo valor deve ser exibido como uma dica de ferramenta para cada nó.

  O valor padrão para cada linguagem de consulta é o seguinte:

  - Para o Gremlin: `T.label`.
  - Para o openCypher: `~labels`.
  - Para o SPARQL: `type`.
- **`--edge-tooltip-property`** (ou **`-te`**): especifica a propriedade cujo valor deve ser exibido como uma dica de ferramenta para cada borda.

  O valor padrão para cada linguagem de consulta é o seguinte:

  - Para o Gremlin: `T.label`.
  - Para o openCypher: `~labels`.
  - Para o SPARQL: `type`.
- **`--label-max-length`** (ou **`-l`**): especifica o tamanho máximo de caracteres de qualquer rótulo de vértice. Padronizado como dez.
- **`--edge-label-max-length`** (ou **`-le`**): especifica o tamanho máximo de caracteres de qualquer rótulo de borda. Padronizado como dez.

  Somente no caso do openCypher, é `--rel-label-max-length` ou `-rel`.

- **`--simulation-duration`** (ou **`-sd`**): especifica a duração máxima da simulação física de visualização. O padrão é 1.500 ms.
- **`--stop-physics`** (ou **`-sp`**): desativa a física de visualização após a estabilização da simulação inicial.

Os valores de propriedade desses argumentos podem consistir em uma única chave de propriedade ou em uma string JSON que pode especificar uma propriedade diferente para cada tipo de rótulo. Uma string JSON só pode ser especificada usando injeção de variável.

A magia de linha **`%seed`**

A magia de linha `%seed` é uma maneira conveniente de adicionar dados ao seu endpoint do Neptune que você pode usar para examinar e experimentar consultas do Gremlin, do openCypher ou do SPARQL. Ela oferece um formulário em que você pode selecionar o modelo de dados que deseja examinar (grafo de propriedades ou RDF) e, depois, escolher entre vários conjuntos de dados de exemplo diferentes oferecidos pelo Neptune.

A magia de linha **`%load`**

A magia de linha `%load` gera um formulário que você pode usar para enviar uma solicitação de carregamento em massa ao Neptune (consulte Comando do carregador do Neptune). O arquivo de origem deve ser um caminho do Amazon S3 na mesma região que o cluster do Neptune.

A magia de linha **`%load_ids`**

A magia de linha `%load_ids` recupera os IDs de carga que foram enviados ao endpoint host do caderno (consulte Parâmetros da solicitação Get-Status do carregador do Neptune). A solicitação tem o seguinte formato:

```
GET https://your-neptune-endpoint:port/loader
```

A magia de linha **`%load_status`**

A magia de linha `%load_status` recupera o status de carregamento de uma tarefa de carga específica que foi enviada ao endpoint host do caderno, especificado pela entrada de linha (consulte Parâmetros da solicitação Get-Status do carregador do Neptune). A solicitação tem o seguinte formato:

```
GET https://your-neptune-endpoint:port/loader?loadId=loadId
```

A magia de linha se assemelha ao seguinte:

```
%load_status load id
```

A magia de linha `%cancel_load`

A magia de linha `%cancel_load` cancela um trabalho de carregamento específico (consulte Trabalho de cancelamento do carregador do Neptune). A solicitação tem o seguinte formato:

```
DELETE https://your-neptune-endpoint:port/loader?loadId=loadId
```

A magia de linha se assemelha ao seguinte:

```
%cancel_load load id
```

A magia de linha `%status`

Recupera informações de status do endpoint host do caderno (%graph_notebook_config mostra o endpoint host).

A magia de linha `%gremlin_status`

Recupera as informações de status da consulta do Gremlin.

A magia de linha `%opencypher_status` (também `%oc_status`)

Recupera o status de uma consulta do opencypher. Essa magia de linha usa os seguintes argumentos opcionais:

- **`--queryId`** ou **`-q`**: determina o ID de uma consulta em execução específica para a qual mostrar o status.
- **`--cancel_query`** ou **`-c`**: cancela uma consulta em execução. Não assume um valor.
- **`--silent`** ou **`-s`**: se `--silent` estiver definido como `true` ao cancelar uma consulta, a consulta em execução será cancelada com um código de resposta HTTP de `200`. Caso contrário, o código de resposta HTTP seria `500`.

- **`--store-to`**: especifica o nome de uma variável na qual armazenar os resultados da consulta.

A magia de linha `%sparql_status`

Recupera as informações de status da consulta do SPARQL.

A magia de linha `%stream_viewer`

A magia de linha `%stream_viewer` exibe uma interface que permite examinar interativamente as entradas registradas nos fluxos do Neptune, se os fluxos estiverem habilitados no cluster do Neptune. Ela aceita os seguintes argumentos opcionais:

- **`language`**: a linguagem de consulta dos dados do fluxo: `gremlin` ou `sparql`. O padrão, se você não fornecer esse argumento, será `gremlin`.
- **`--limit`**: especifica o número máximo de entradas de fluxo a serem exibidas por página. O valor padrão, se você não fornecer esse argumento, será `10`.

> **Note**
>
> A magia de linha `%stream_viewer` é totalmente aceita somente nas versões 1.0.5.1 e anteriores do mecanismo.

A magia de linha `%graph_notebook_config`

Essa magia de linha exibe um objeto JSON que contém a configuração que o caderno está usando para se comunicar com o Neptune. A configuração inclui:

- `host`: o endpoint ao qual se conectar e emitir comandos.
- `port`: a porta usada ao emitir comandos para o Neptune. O padrão é `8182`.
- `auth_mode`: o modo de autenticação a ser usado ao emitir comandos para o Neptune. Deve ser `IAM` caso você esteja se conectando a um cluster que tenha a autenticação do IAM habilitada. Caso contrário, `DEFAULT`.
- `load_from_s3_arn`: especifica um ARN do Amazon S3 para a magia `%load` a ser usada. Se esse valor estiver vazio, o ARN deverá ser especificado no comando `%load`.

- `ssl`: um valor booliano que indica se você deve ou não se conectar ao Neptune usando TLS. O valor padrão é `true`.
- `aws_region`: a região em que esse caderno é implantado. Essas informações são usadas para autenticação do IAM e para solicitações `%load`.

É possível alterar a configuração copiando a saída `%graph_notebook_config` em uma nova célula e fazendo alterações nela. Depois, se você executar a magia de célula [%%graph_notebook_config] na nova célula, a configuração será alterada adequadamente.

A magia de linha **`%graph_notebook_host`**

Define a entrada de linha como host do caderno.

A magia de linha **`%graph_notebook_version`**

A magia de linha `%graph_notebook_version` gera o número de lançamento do caderno da bancada de trabalho do Neptune. Por exemplo, a visualização do grafo foi introduzida na versão `1.27`.

A magia de linha **`%graph_notebook_vis_options`**

A magia de linha `%graph_notebook_vis_options` exibe as configurações de visualização atuais que o caderno está usando. Essas opções são explicadas na documentação do vis.js.

É possível modificar essas configurações copiando a saída em uma nova célula, fazendo as alterações desejadas e, depois, executando a magia de célula `%%graph_notebook_vis_options` na célula.

Para restaurar os valores padrão das configurações de visualização, você pode executar a magia de linha `%graph_notebook_vis_options` com um parâmetro `reset`. Isso redefine todas as configurações de visualização:

```
%graph_notebook_vis_options reset
```

A magia de linha **`%statistics`**

A magia de linha `%statistics` é usada para recuperar ou gerenciar estatísticas do mecanismo DFE (consulte Gerenciar estatísticas a serem utilizadas pelo DFE do Neptune). Essa magia também pode ser usada para recuperar um resumo do grafo.

Ela aceita os seguintes parâmetros:

- **`--language`**: a linguagem de consulta do endpoint de estatísticas: `propertygraph` (ou pg) ou `rdf`.

  Se não for fornecido, o padrão será `propertygraph`.
- **`--mode`** (ou **`-m`**): especifica o tipo de solicitação ou ação a ser enviada: uma de `status`, `disableAutoCompute`, `enableAutoCompute`, `refresh`, `delete`, `detailed` ou `basic`).

  Se não for fornecido, o padrão será `status`, a menos que `--summary` seja especificado; nesse caso, o padrão será `basic`.
- **`--summary`**: recupera o resumo do grafo do endpoint do resumo de estatísticas da linguagem selecionada.
- **`--silent`**: se estiver presente, nenhuma saída será exibida após a conclusão da consulta.
- **`--store-to`**: usado para especificar uma variável na qual armazenar os resultados da consulta.

A magia de linha **`%summary`**

A magia de linha `%summary` é usada para recuperar informações de resumo do grafo. Ela está disponível a partir da versão do mecanismo do Neptune `1.2.1.0`.

Ela aceita os seguintes parâmetros:

- **`--language`**: a linguagem de consulta do endpoint de estatísticas: `propertygraph` (ou pg) ou `rdf`.

  Se não for fornecido, o padrão será `propertygraph`.
- **`--detailed`**: ativa ou desativa a exibição dos campos de estruturas na saída.

  Se não for fornecido, o padrão será o modo de exibição de resumo `basic`.
- **`--silent`**: se estiver presente, nenhuma saída será exibida após a conclusão da consulta.
- **`--store-to`**: usado para especificar uma variável na qual armazenar os resultados da consulta.

A magia de célula **`%%graph_notebook_config`**

A magia de célula `%%graph_notebook_config` usa um objeto JSON que contém informações de configuração para modificar as configurações que o caderno está usando para se comunicar

com o Neptune, se possível. A configuração assume a mesma forma gerada pela magia de linha %graph_notebook_config.

Por exemplo: .

```
%%graph_notebook_config
{
  "host": "my-new-cluster-endpoint.amazon.com",
  "port": 8182,
  "auth_mode": "DEFAULT",
  "load_from_s3_arn": "",
  "ssl": true,
  "aws_region": "us-east-1"
}
```

A magia de célula **%%sparql**

A magia de célula `%%sparql` emite uma consulta do SPARQL para o endpoint do Neptune. Ela aceita a seguinte entrada de linha opcional:

- **`-h`** ou **`--help`**: gera o texto de ajuda sobre esses parâmetros.
- **`--path`**: adiciona um prefixo a um caminho para o endpoint SPARQL. Por exemplo, se você especificar `--path "abc/def"`, o endpoint chamado será *`host`*`:`*`port`*`/abc/def`.
- **`--expand-all`**: é uma dica de visualização de consulta que diz ao visualizador para incluir todos os resultados `?s ?p ?o` no diagrama do grafo, independentemente do tipo de vinculação.

  Por padrão, uma visualização do SPARQL inclui apenas padrões triplos em que o`?` é um `uri` ou um `bnode` (nó em branco). Todos os outros tipos de vinculação **`?o`**, como strings literais ou números inteiros, são tratados como propriedades do nó **`?s`** que podem ser visualizadas usando o painel Detalhes na guia Grafo.

  Em vez disso, use a dica de consulta `--expand-all` quando quiser incluir valores literais como vértices na visualização.

  Não combine essa dica de visualização com os parâmetros de explicação, pois as consultas de explicação não são visualizadas.
- **`--explain-type`**: utilizado para especificar o modo de explicação a ser usado (um dos seguintes: `dynamic`, `static` ou `details`).

- **`--explain-format`**: usado para especificar o formato de resposta para uma consulta de explicação (`text/csv` ou `text/html`).
- --store-to: usado para especificar uma variável na qual armazenar os resultados da consulta.

Exemplo de consulta `explain`:

```
%%sparql explain

SELECT * WHERE {?s ?p ?o} LIMIT 10
```

Exemplo de consulta de visualização com um parâmetro de dica de visualização `--expand-all` (consulte Visualização do SPARQL):

```
%%sparql --expand-all

SELECT * WHERE {?s ?p ?o} LIMIT 10
```

A magia de célula **`%%gremlin`**

A magia `%%gremlin` celular emite uma consulta Gremlin para o endpoint de Neptune usando. WebSocket Ela aceita uma entrada de linha opcional para alternar para o modo `explain` do Gremlin /> ou API `profile` do Gremlin e uma entrada de dica de visualização opcional separada para modificar o comportamento da saída de visualização (consulte Visualização do Gremlin).

Exemplo de consulta `explain`:

```
%%gremlin explain

g.V().limit(10)
```

Exemplo de consulta `profile`:

```
%%gremlin profile

g.V().limit(10)
```

Exemplo de consulta de visualização com um parâmetro de dica de visualização:

```
%%gremlin -p v,outv
```

```
g.V().out().limit(10)
```

Parâmetros opcionais para consultas **`%%gremlin profile`**

- **`--chop`**: especifica o tamanho máximo da string de resultados do perfil. O valor padrão, se você não fornecer esse argumento, será 250.
- **`--serializer`**: especifica o serializador a ser usado para os resultados. Os valores permitidos são qualquer um dos valores de enumeração “Serializadores” do tipo MIME ou TinkerPop driver válidos. O valor padrão, se você não fornecer esse argumento, será `application.json`.
- **`--no-results`**: exibe somente a contagem de resultados. Se não for usado, todos os resultados da consulta serão exibidos no relatório do perfil por padrão.
- **`--indexOps`**: mostra um relatório detalhado de todas as operações do índice.

A magia de célula **`%%opencypher`** (também **`%%oc`**)

A magia de célula `%%opencypher` (que também tem a forma abreviada `%%oc`) emite uma consulta do openCypher para o endpoint do Neptune. Ela aceita os seguintes argumentos de entrada de linha opcionais:

- mode: o modo de consulta: `query` ou `bolt`. O valor padrão, se você não fornecer esse argumento, será `query`.
- **`--group-by`** ou **`-g`**: especifica a propriedade usada para agrupar nós. Por exemplo, `code, ~id`. O valor padrão, se você não fornecer esse argumento, será `~labels`.
- **`--ignore-groups`**: se estiverem presentes, todas as opções de agrupamento serão ignoradas.
- **`--display-propery`** ou **`-d`**: especifica a propriedade cujo valor deve ser exibido para cada vértice. O valor padrão, se você não fornecer esse argumento, será `~labels`.
- **`--edge-display-propery`** ou **`-de`**: especifica a propriedade cujo valor deve ser exibido para cada borda. O valor padrão, se você não fornecer esse argumento, será `~labels`.
- **`--label-max-length`** ou **`-l`**: especifica o número máximo de caracteres de um rótulo de vértice a ser exibido. O valor padrão, se você não fornecer esse argumento, será `10`.
- **`--store-to`** ou **`-s`**: especifica o nome de uma variável na qual armazenar os resultados da consulta.
- **`--plan-cache`** ou **`-pc`**: especifica o modo de cache do plano a ser usado. O valor padrão é`auto`. (*o plan-cache só está disponível para o Neptune Analytics)

- **`--query-timeout`** ou **`-qt`**: especifica o tempo limite máximo da consulta em milissegundos. O valor padrão é `1800000`.
- **`--query-parameters`** ou **`qp`**: definições de parâmetros a serem aplicadas à consulta. Essa opção pode aceitar um único nome de variável ou uma representação de string do mapa.

  Exemplo de uso de **`--query-parameters`**

  1. Defina um mapa dos parâmetros do openCypher em uma célula do bloco de anotações.

     ```
     params = '''{
      "name":"john",
      "age": 20,
     }'''
     ```

  2. Passe os parâmetros para `--query-parameters` em outra célula com `%%oc`.

     ```
     %%oc --query-parameters params

     MATCH (n {name: $name, age: $age})
     RETURN n
     ```

- --explain-type — Usado para especificar o modo de explicação a ser usado (um dos seguintes: dinâmico, estático ou detalhado).

A magia de célula **`%%graph_notebook_vis_options`**

A magia de célula `%%graph_notebook_vis_options` permite que você defina opções de visualização para o caderno. É possível copiar as configurações geradas pela magia de linha `%graph-notebook-vis-options` em uma nova célula, fazer alterações nelas e usar a magia de célula `%%graph_notebook_vis_options` para definir os novos valores.

Essas opções são explicadas na documentação do vis.js.

Para restaurar os valores padrão das configurações de visualização, você pode executar a magia de linha `%graph_notebook_vis_options` com um parâmetro `reset`. Isso redefine todas as configurações de visualização:

```
%graph_notebook_vis_options reset
```

A magia de linha **`%neptune_ml`**

É possível usar a magia de linha `%neptune_ml` para iniciar e gerenciar várias operações do Neptune ML.

> **Note**
>
> Também é possível iniciar e gerenciar algumas operações do Neptune ML usando a magia de célula %%neptune_ml.

- **`%neptune_ml export start`**: inicia um novo trabalho de exportação.

  Parâmetros

  - **`--export-url`** *`exporter-endpoint`*: (opcional) o endpoint do Amazon API Gateway em que o exportador pode ser chamado.
  - **`--export-iam`**: (opcional) sinalizador indicando que as solicitações para o URL de exportação devem ser assinadas usando SigV4.
  - **`--export-no-ssl`**: (opcional) sinalizador indicando que o SSL não deve ser usado ao se conectar ao exportador.
  - **`--wait`**: (opcional) sinalizador indicando que a operação deve esperar até que a exportação seja concluída.
  - **`--wait-interval`***`interval-to-wait`*— (opcional) Define o tempo, em segundos, entre as verificações de status de exportação (Padrão: 60).
  - **`--wait-timeout`** *`timeout-seconds`*: (opcional) define o tempo, em segundos, para aguardar a conclusão do trabalho de exportação antes de exibir o status mais recente (padrão: 3.600).
  - **`--store-to`***`location-to-store-result`*— (opcional) A variável na qual armazenar o resultado da exportação. Se `--wait` for especificado, o status final será armazenado.
- **`%neptune_ml export status`**: recupera o status de um trabalho de exportação.

  Parâmetros

  - **`--job-id`** *`ID do trabalho de exportação`*: o ID do trabalho de exportação para o qual recuperar o status.
  - **`--export-url`** *`exporter-endpoint`*: (opcional) o endpoint do Amazon API Gateway em que o exportador pode ser chamado.

- **`--export-iam`**: (opcional) sinalizador indicando que as solicitações para o URL de exportação devem ser assinadas usando SigV4.
- **`--export-no-ssl`**: (opcional) sinalizador indicando que o SSL não deve ser usado ao se conectar ao exportador.
- **`--wait`**: (opcional) sinalizador indicando que a operação deve esperar até que a exportação seja concluída.
- **`--wait-interval`*`interval-to-wait`*** — (opcional) Define o tempo, em segundos, entre as verificações de status de exportação (Padrão: 60).
- **`--wait-timeout`** *`timeout-seconds`*: (opcional) define o tempo, em segundos, para aguardar a conclusão do trabalho de exportação antes de exibir o status mais recente (padrão: 3.600).
- **`--store-to`*`location-to-store-result`*** — (opcional) A variável na qual armazenar o resultado da exportação. Se `--wait` for especificado, o status final será armazenado.

- **`%neptune_ml dataprocessing start`**: inicia a etapa de processamento de dados do Neptune ML.

  Parâmetros

  - **`--job-id`** *`ID desse trabalho`*: (opcional) ID a ser atribuído a esse trabalho.
  - **`--s3-input-uri`** *`URI do S3`*: (opcional) o URI do S3 no qual se encontra a entrada desse trabalho de processamento de dados.
  - **`--config-file-name`** *`nome do arquivo`*: (opcional) nome do arquivo de configuração desse trabalho de processamento de dados.
  - **`--store-to`*`location-to-store-result`*** — (opcional) A variável na qual armazenar o resultado do processamento de dados.
  - **`--instance-type`** *`(tipo de instância)`*: (opcional) O tamanho da instância a ser usada para esse trabalho de processamento de dados.
  - **`--wait`**: (opcional) sinalizador indicando que a operação deve esperar até que o processamento de dados seja concluído.
  - **`--wait-interval`*`interval-to-wait`*** — (opcional) Define o tempo, em segundos, entre as verificações do status do processamento de dados (Padrão: 60).
  - **`--wait-timeout`** *`timeout-seconds`*: (opcional) define o tempo, em segundos, para aguardar a conclusão do trabalho de processamento de dados antes de exibir o status mais recente (padrão: 3.600).

- **`%neptune_ml dataprocessing status`**: recupera o status de uma tarefa de processamento de dados.

  Parâmetros

  - **`--job-id`** *`ID do trabalho`*: o ID do trabalho para o qual recuperar o status.
  - **`--store-to`** *`tipo de instância`*: (opcional) a variável na qual armazenar o resultado do treinamento do modelo.
  - **`--wait`**: (opcional) sinalizador indicando que a operação deve aguardar até que o treinamento de modelos seja concluído.
  - **`--wait-interval`*`interval-to-wait`*** — (opcional) Define o tempo, em segundos, entre as verificações de status do treinamento do modelo (Padrão: 60).
  - **`--wait-timeout`** *`timeout-seconds`*: (opcional) define o tempo, em segundos, para aguardar a conclusão do trabalho de processamento de dados antes de exibir o status mais recente (padrão: 3.600).
- **`%neptune_ml training start`**: inicia o processo de treinamento de modelos do Neptune ML.

  Parâmetros

  - **`--job-id`** *`ID desse trabalho`*: (opcional) ID a ser atribuído a esse trabalho.
  - **`--data-processing-id`** *`ID do trabalho de processamento de dados`*: (opcional) ID do trabalho de processamento de dados que criou os artefatos a serem usados no treinamento.
  - **`--s3-output-uri`** *`URI do S3`*: (opcional) o URI do S3 no qual armazenar a saída desse trabalho de treinamento de modelos.
  - **`--instance-type`** *`(tipo de instância)`*: (opcional) o tamanho da instância a ser usada para esse trabalho de treinamento de modelos.
  - **`--store-to`*`location-to-store-result`*** — (opcional) A variável na qual armazenar o resultado do treinamento do modelo.
  - **`--wait`**: (opcional) sinalizador indicando que a operação deve aguardar até que o treinamento de modelos seja concluído.
  - **`--wait-interval`*`interval-to-wait`*** — (opcional) Define o tempo, em segundos, entre as verificações de status do treinamento do modelo (Padrão: 60).
  - **`--wait-timeout`** *`timeout-seconds`*: (opcional) define o tempo, em segundos, para aguardar a conclusão do trabalho de treinamento de modelos antes de exibir o status mais recente (padrão: 3.600).

- **`%neptune_ml training status`**: recupera o status de um trabalho de treinamento de modelos do Neptune ML.

  Parâmetros

  - **`--job-id`** *`ID do trabalho`*: o ID do trabalho para o qual recuperar o status.
  - **`--store-to`** *`tipo de instância`*: (opcional) a variável na qual armazenar o resultado do status.
  - **`--wait`**: (opcional) sinalizador indicando que a operação deve aguardar até que o treinamento de modelos seja concluído.
  - **`--wait-interval`*`interval-to-wait`*** — (opcional) Define o tempo, em segundos, entre as verificações de status do treinamento do modelo (Padrão: 60).
  - **`--wait-timeout`** *`timeout-seconds`*: (opcional) define o tempo, em segundos, para aguardar a conclusão do trabalho de processamento de dados antes de exibir o status mais recente (padrão: 3.600).
- **`%neptune_ml endpoint create`**: cria um endpoint de consulta para um modelo do Neptune ML.

  Parâmetros

  - **`--job-id`** *`ID desse trabalho`*: (opcional) ID a ser atribuído a esse trabalho.
  - **`--model-job-id`** *`ID do trabalho de treinamento de modelos`*: (opcional) ID do trabalho de treinamento de modelos para o qual criar um endpoint de consulta.
  - **`--instance-type`** *`(tipo de instância)`*: (opcional) o tamanho da instância a ser usada para o endpoint de consulta.
  - **`--store-to`*`location-to-store-result`*** — (opcional) A variável na qual armazenar o resultado da criação do endpoint.
  - **`--wait`**: (opcional) sinalizador indicando que a operação deve aguardar até que a criação do endpoint seja concluída.
  - **`--wait-interval`*`interval-to-wait`*** — (opcional) Define o tempo, em segundos, entre as verificações de status (Padrão: 60).
  - **`--wait-timeout`** *`timeout-seconds`*: (opcional) define o tempo, em segundos, para aguardar a conclusão do trabalho de criação do endpoint antes de exibir o status mais recente (padrão: 3.600).
- **`%neptune_ml endpoint status`**: recupera o status de um endpoint de consulta do Neptune ML.

Parâmetros

- **`--job-id`** *`ID de criação de endpoint`*: (opcional) ID de um trabalho de criação de endpoint para o qual relatar o status.
- **`--store-to`***`location-to-store-result`*— (opcional) A variável na qual armazenar o resultado do status.
- **`--wait`**: (opcional) sinalizador indicando que a operação deve aguardar até que a criação do endpoint seja concluída.
- **`--wait-interval`***`interval-to-wait`*— (opcional) Define o tempo, em segundos, entre as verificações de status (Padrão: 60).
- **`--wait-timeout`** *`timeout-seconds`*: (opcional) define o tempo, em segundos, para aguardar a conclusão do trabalho de criação do endpoint antes de exibir o status mais recente (padrão: 3.600).

A magia de célula `%%neptune_ml`

A magia de célula `%%neptune_ml` ignora as entradas de linha, como `--job-id` ou `--export-url`. Em vez disso, ela permite que você forneça essas entradas e outras dentro do corpo da célula.

Também é possível salvar essas entradas em outra célula, atribuída a uma variável Jupyter, e depois injetá-las no corpo da célula usando essa variável. Dessa forma, é possível usar essas entradas repetidamente sem precisar inseri-las novamente todas as vezes.

Isso só funcionará se a variável de injeção for o único conteúdo da célula. Você não pode usar várias variáveis em uma célula nem uma combinação de texto e uma variável.

Por exemplo, a magia de célula `%%neptune_ml export start` pode consumir um documento JSON no corpo da célula que contém todos os parâmetros descritos em Parâmetros usados para controlar o processo de exportação do Neptune.

No caderno Neptune-ML-01-Introduction-to-Node-Classification-Gremlin, em Configuring Features na seção Export the data and model configuration, é possível ver como a célula a seguir contém os parâmetros de exportação em um documento atribuído a uma variável do Jupyter chamada `export-params`.

```
export_params = {
  "command": "export-pg",
```

```
  "params": {
    "endpoint": neptune_ml.get_host(),
    "profile": "neptune_ml",
    "useIamAuth": neptune_ml.get_iam(),
    "cloneCluster": False
   },
  "outputS3Path": f'{s3_bucket_uri}/neptune-export',
  "additionalParams": {
    "neptune_ml": {
      "targets": [
        {
          "node": "movie",
          "property": "genre"
        }
      ],
      "features": [
        {
          "node": "movie",
          "property": "title",
          "type": "word2vec"
        },
        {
          "node": "user",
          "property": "age",
          "type": "bucket_numerical",
          "range" : [1, 100],
          "num_buckets": 10
        }
      ]
    }
  },
  "jobSize": "medium"}
```

Quando você executa essa célula, o Jupyter salva o documento de parâmetros com esse nome. Depois, é possível usar `${export_params}` para injetar o documento JSON no corpo de um `%%neptune_ml export start cell` da seguinte forma:

```
%%neptune_ml export start --export-url {neptune_ml.get_export_service_host()} --export-
iam --wait --store-to export_results

${export_params}
```

Formas disponíveis da magia de célula `%%neptune_ml`

A magia de célula `%%neptune_ml` pode ser usada das seguintes formas:

- **`%%neptune_ml export start`**: inicia um processo de exportação do Neptune ML.
- **`%%neptune_ml dataprocessing start`**: inicia um trabalho de processamento de dados do Neptune ML.
- **`%%neptune_ml training start`**: inicia um trabalho de treinamento de modelos do Neptune ML.
- **`%%neptune_ml endpoint create`**: cria um endpoint de consulta do Neptune ML para um modelo.

Visualização de grafos na bancada de trabalho do Neptune

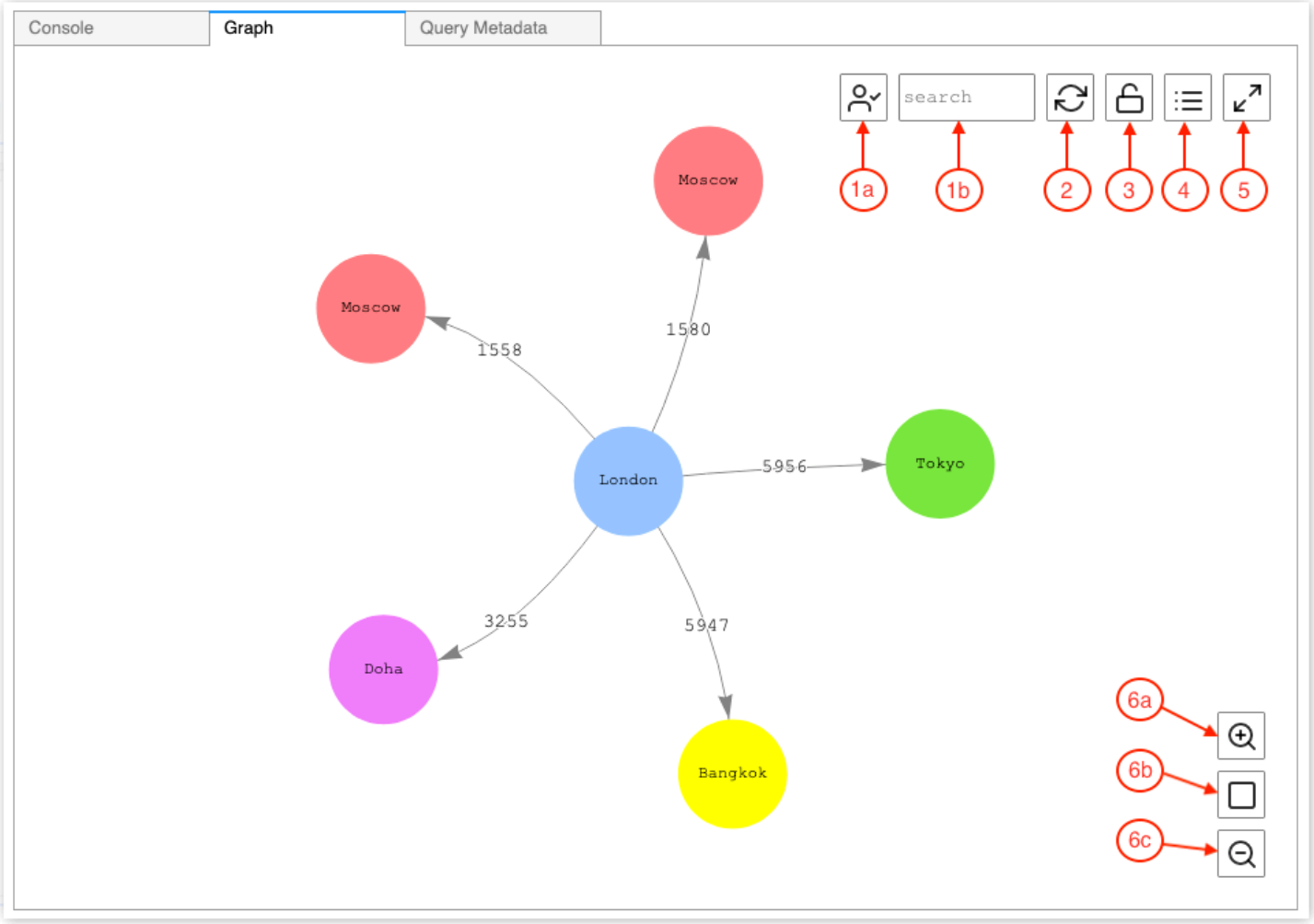
Em muitos casos, a bancada de trabalho do Neptune pode criar um diagrama visual dos resultados da consulta, bem como exibi-los em formato tabular. A visualização do grafo estará disponível na guia Grafo nos resultados da consulta sempre que a visualização for possível.

Além dos recursos de visualização integrados descritos aqui, também é possível usar ferramentas de visualização mais avançadas com os blocos de anotações de grafos Neptune.

> **Note**
>
> Para ter acesso às funcionalidades e às correções adicionadas recentemente aos blocos de anotações que você já está usando, primeiro pare e reinicie a instância do bloco de anotações.

Visão geral da interface da guia Grafo

Esse diagrama identifica os elementos da interface do usuário presentes na guia Grafo:



1. Pesquisa de grafos
   a. Alternar UUID: alterna a inclusão de valores de propriedade de ID na pesquisa de grafos. Por padrão, a inclusão de ID é habilitada. Se desabilitada, as correspondências nas propriedades de ID, incluindo propriedades de borda que fazem referência a IDs de nós, não ocasionam o destaque do elemento.
   b. Campo de texto de pesquisa: destaca todos os valores de propriedades de vértice e borda que contêm a string de texto que você especifica aqui.
2. Redefinição do grafo: executa novamente a simulação de física do grafo e define o zoom para ajustar o grafo na janela.
3. Alternar física do grafo: alterna a execução da simulação da física do grafo. A física é habilitada por padrão, permitindo que o grafo mude dinamicamente. Se desabilitado, os vértices permanecem bloqueados na posição quando outros vértices são movidos.

4. Visualização de detalhes: quando uma borda ou um nó é selecionado, isso exibe uma lista das chaves e valores das propriedades do elemento, se disponíveis nos resultados da consulta.
5. Visualização em tela cheia: expande a janela da guia do grafo para caber na tela. Clicar novamente minimiza a guia do grafo.
6. Opções de zoom
   a. Aumentar zoom
   b. Redefinição de zoom: define o zoom para caber todos os vértices na janela da guia do grafo.
   c. Diminuir zoom

Visualizar resultados da consulta do Gremlin

A bancada de trabalho do Neptune cria uma visualização dos resultados da consulta para qualquer consulta do Gremlin que exiba um `path`. Para ver a visualização, selecione a guia Grafo à direita da guia Console, abaixo da consulta, depois de executá-la.

É possível usar dicas de visualização de consulta para controlar como o visualizador faz a diagramação da saída da consulta. Essas dicas seguem a magia de célula `%%gremlin` e são precedidas pelo nome do parâmetro `--path-pattern` (ou a forma abreviada, `-p`):

```
%%gremlin -p comma-separated hints
```

Também é possível usar o sinalizador `--group-by` (ou `-g`) para especificar uma propriedade dos vértices pela qual agrupá-los. Isso permite especificar uma cor ou um ícone para diferentes grupos de vértices.

Os nomes das dicas refletem as etapas do Gremlin comumente usadas ao percorrer entre vértices e se comportam adequadamente. Várias dicas podem ser usadas em combinação, separadas por vírgulas, sem espaços entre elas. As dicas usadas devem combinar com as etapas correspondentes do Gremlin na consulta que está sendo visualizada. Exemplo:

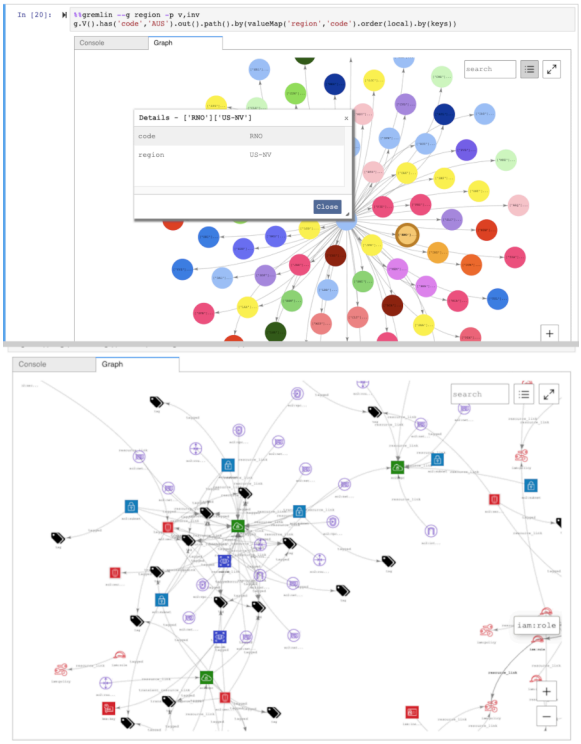
```
%%gremlin -p v,oute,inv
g.V().hasLabel('airport').outE().inV().path().by('code').by('dist').limit(5)
```

As dicas de visualização disponíveis são as seguintes:

```
v
inv
```

```
outv
e
ine
oute
```

Estes são alguns exemplos de visualização de grafos usando grupos:



Visualizar resultados da consulta do SPARQL

A bancada de trabalho do Neptune cria uma visualização dos resultados da consulta para qualquer consulta do SPARQL que tenha um dos seguintes formatos:

- `SELECT ?subject ?predicate ?object`
- `SELECT ?s ?p ?o`

Para ver a visualização, selecione a guia Grafo à direita da guia Tabela, abaixo da consulta, depois de executá-la.

Por padrão, uma visualização do SPARQL inclui apenas padrões triplos em que o `?` é um `uri` ou um `bnode` (nó em branco). Todos os outros tipos de vinculação `?o`, como strings literais ou números inteiros, são tratados como propriedades do nó `?s` que podem ser visualizadas no painel Detalhes na guia Grafo.

Em muitos casos, no entanto, talvez você queira incluir valores literais como vértices na visualização. Para isso, use a dica de consulta `--expand-all` após a magia de célula `%%sparql`:

```
%%sparql --expand-all
```

Isso informa ao visualizador para incluir todos os resultados `?s ?p ?o` no diagrama do grafo, independentemente do tipo de vinculação.

É possível ver essa dica usada em todo o bloco de anotações `Air-Routes-SPARQL.ipynb` e pode experimentar executando as consultas com e sem a dica para ver a diferença que isso faz na visualização.

Acessar blocos de anotações de tutoriais de visualização na bancada de trabalho do Neptune

Os dois blocos de anotações de tutoriais de visualização que vêm com a bancada de trabalho do Neptune fornecem muitos exemplos em Gremlin e em SPARQL de como consultar dados de grafos de forma eficaz e visualizar os resultados.

Acessar os blocos de anotações de visualização

1. No painel de navegação à esquerda, escolha o botão Abrir bloco de anotações à direita.
2. Depois de abrir a bancada de trabalho do Neptune, executando o Jupyter, você verá uma pasta do Neptune no nível superior. Selecione-a para abrir a pasta.
3. No próximo nível, há uma pasta chamada 02-Visualization. Abra esta pasta. Dentro, há vários blocos de anotações que orientam você sobre diferentes maneiras de consultar seus dados de grafos, no Gremlin e no SPARQL, e como visualizar os resultados da consulta:

   - Air-Routes-Gremlin
   - Air-Routes-SPARQL
   - Blog Visualização da bancada de trabalho
   - EPL-Gremlin
   - EPL-SPARQL

   Selecione um bloco de anotações para fazer experiências com as consultas que ele contém.

Configurar o Neptune

Bem-vindo ao Amazon Neptune. Esta seção ajuda você a criar um cluster de banco de dados do Neptune e encontrar o que você está procurando na documentação do Neptune.

> **Note**
>
> Para arquiteturas de referência de banco de dados AWS gráficos e arquiteturas de implantação de referência, consulte Amazon Neptune Resources. Esses recursos podem ajudar a embasar suas escolhas sobre modelos de dados de gráficos e linguagens de consulta e acelerar o processo de desenvolvimento.

Tópicos

Escolher o tipo certo de instância de banco de dados do Neptune

O Amazon Neptune oferece vários tamanhos e famílias de instâncias diferentes que oferecem recursos distintos, adequados a diferentes workloads de grafos. Esta seção tem como objetivo ajudar você a escolher o melhor tipo de instância para suas necessidades.

Para conhecer os preços de cada tipo de instância nessas famílias, consulte a página Preços do Neptune.

Visão geral da alocação de recursos da instância

Cada tipo e tamanho de instância do Amazon EC2 usados no Neptune oferece uma quantidade definida de computação (vCPUs) e memória do sistema. O armazenamento primário do Neptune é externo às instâncias de banco de dados em um cluster, o que permite que a capacidade de computação e armazenamento sejam escalados de forma independente um do outro.

Esta seção se concentra em como os recursos computacionais podem ser escalados e nas diferenças entre cada uma das várias famílias de instâncias.

Em todas as famílias de instâncias, os recursos de vCPU são alocados para oferecer compatibilidade com dois (2) threads de execução de consultas por vCPU. Essa compatibilidade é determinada pelo tamanho da instância. Ao determinar o tamanho adequado de uma instância de banco de dados Neptune específica, é necessário considerar a possível simultaneidade da aplicação e a latência média das consultas. Você pode estimar o número de vCPUs necessárias da seguinte forma, em que a latência é medida como a latência média da consulta em segundos e a simultaneidade é medida como o número de destino de consultas por segundo:

$$vCPUs = \frac{latency \times concurrency}{2}$$

> **Note**
>
> As consultas SPARQL, openCypher e de leitura do Gremlin que usam o mecanismo de consulta DFE podem, em determinadas circunstâncias, usar mais de um thread de execução por consulta. Ao dimensionar inicialmente o cluster de banco de dados, comece com a suposição de que cada consulta consumirá um único thread de execução por execução e aumente a escala verticalmente se você observar uma pressão de retorno na fila de consultas. Isso pode ser observado usando as `/sparql/status` APIs `/gremlin/status/oc/status`,, ou também pode ser observado usando a `MainRequestsPendingRequestsQueue` CloudWatch métrica.

A memória do sistema em cada instância é dividida em duas alocações principais: cache do grupo do buffer e memória do thread de execução da consulta.

Aproximadamente dois terços da memória disponível em uma instância são alocados para o cache do grupo de buffer. O cache do grupo de buffer é usado para armazenar em cache os componentes do grafo usados mais recentemente para acesso mais rápido às consultas que acessam repetidamente esses componentes. Instâncias com uma quantidade maior de memória do sistema têm caches maiores de grupo de buffer que podem armazenar mais do grafo localmente. Um usuário pode ajustar a quantidade adequada de cache do pool de buffer monitorando as métricas de acertos e erros do cache de buffer disponíveis em. CloudWatch

Convém aumentar o tamanho da instância se a taxa de acerto do cache cair abaixo de 99,9% por um período consistente. Isso sugere que o grupo de buffer não é grande o suficiente e que o mecanismo precisa buscar dados do volume de armazenamento subjacente com frequência maior do que é eficiente.

O terço restante da memória do sistema é distribuído uniformemente entre os threads de execução de consulta, com alguma memória restante para o sistema operacional e um pequeno grupo dinâmico para os threads usarem conforme necessário. A memória disponível para cada thread aumenta ligeiramente de um tamanho de instância para o próximo até um tipo de instância `8xl`, tamanho em que a memória alocada por thread atinge o máximo.

A hora de adicionar mais memória de thread é quando você encontra um `OutOfMemoryException` (OOM). As exceções de OOM ocorrem quando um thread precisa de mais do que a memória máxima alocada para ele (não é o mesmo que a instância inteira ficar sem memória).

Tipos de instância **`t3`** e **`t4g`**

A família de instâncias `t3` e `t4g` oferece uma opção de baixo custo para começar a usar um banco de dados de grafos e também para desenvolvimento e teste iniciais. Essas instâncias são elegíveis para a oferta de nível gratuito do Neptune, que permite que novos clientes usem o Neptune sem nenhum custo nas primeiras 750 horas de instância usadas em uma conta AWS independente ou acumuladas AWS em uma organização com faturamento consolidado (conta do pagador).

As instâncias `t3` e `t4g` são oferecidas somente na configuração de tamanho médio (`t3.medium` e `t4g.medium`).

Elas não se destinam ao uso em um ambiente de produção.

Como essas instâncias têm recursos muito restritos, elas não são recomendadas para testar o tempo de execução da consulta nem o desempenho geral do banco de dados. Para avaliar o desempenho da consulta, faça a atualização para as outras famílias de instância.

Família **r4** de tipos de instância

OBSOLETA: a família `r4` era oferecida quando o Neptune foi lançado em 2018, mas agora os tipos de instância mais novos oferecem preço/desempenho muito melhores. A partir da versão 1.1.0.0 do mecanismo, o Neptune não é mais compatível com tipos de instância `r4`.

Família **r5** de tipos de instância

A família `r5` contém tipos de instância otimizada para memória que funcionam bem para a maioria dos casos de uso de grafos. A família `r5` contém tipos de instância de `r5.large` a `r5.24xlarge`. Elas são escaladas linearmente no desempenho computacional à medida que você aumenta de tamanho. Por exemplo, uma `r5.xlarge` (4 vCPUs e 32 GiB de memória) tem o dobro das vCPUs e a memória de uma `r5.large` (2 vCPUs e 16 GiB de memória), e uma `r5.2xlarge` (8 vCPUs e 64 GiB de memória) tem o dobro das vCPUs e da memória de uma `r5.xlarge`. Você pode esperar que o desempenho da consulta seja escalado diretamente com a capacidade computacional até o tipo de instância `r5.12xlarge`.

A família de instâncias `r5` tem uma arquitetura de CPU Intel de 2 soquetes. A `r5.12xlarge` e tipos menores usam um único soquete e a memória do sistema pertencente a esse processador de soquete único. Os tipos `r5.16xlarge` e `r5.24xlarge` usam os dois soquetes e a memória disponível. Como há alguma sobrecarga de gerenciamento de memória necessária entre dois processadores físicos em uma arquitetura de 2 soquetes, os ganhos de desempenho ao aumentar a escala verticalmente de um tipo de instância `r5.12xlarge` para `r5.16xlarge` ou `r5.24xlarge` instância não são tão lineares quanto os obtidos ao aumentar a escala verticalmente em tamanhos menores.

Família **r5d** de tipos de instância

O Neptune tem um atributo de cache de pesquisa que pode ser usado para melhorar o desempenho de consultas que precisam buscar e gerar um grande número de valores de propriedades e literais. Esse atributo é usado principalmente por clientes com consultas que precisam gerar vários atributos. O cache de pesquisa aumenta o desempenho dessas consultas ao buscar esses valores de atributos localmente, em vez de pesquisar cada um deles repetidamente no armazenamento indexado do Neptune.

O cache de pesquisa é implementado usando um volume EBS conectado ao NVMe em um tipo de instância `r5d`. Ele é habilitado usando o grupo de parâmetros de um cluster. À medida que os dados são obtidos do armazenamento indexado do Neptune, os valores das propriedades e os literais RDF são armazenados em cache nesse volume NVMe.

Se você não precisar do atributo de cache de pesquisa, use um tipo de instância `r5` padrão em vez de um `r5d`, para evitar o custo mais alto da `r5d`.

A família `r5d` tem tipos de instância nos mesmos tamanhos da família `r5`, de `r5d.large` a `r5d.24xlarge`.

Família **`r6g`** de tipos de instância

AWS desenvolveu seu próprio processador baseado em ARM chamado Graviton, que oferece melhor preço/desempenho do que os equivalentes Intel e AMD. A família `r6g` usa o processador Graviton2. Em nossos testes, o processador Graviton2 oferece desempenho de 10% a 20% melhor para consultas de grafos no estilo OLTP (restritas). No entanto, consultas OLAP maiores podem ter um desempenho um pouco inferior com os processadores Graviton2 do que com as com processadores Intel, devido ao desempenho um pouco inferior de paginação na memória.

Também é importante observar que a família `r6g` tem uma arquitetura de soquete único, o que significa que o desempenho é escalado linearmente com a capacidade computacional de uma `r6g.large` para uma `r6g.16xlarge` (o maior tipo da família).

Família **`r6i`** de tipos de instância

As instâncias R6i da Amazon são alimentadas por processadores escaláveis Intel Xeon de 3ª geração (codinome Ice Lake) e são ideais para workloads com uso intenso de memória. Como regra, elas oferecem desempenho de preço de computação até 15% melhor e largura de banda de memória até 20% maior por vCPU do que tipos de instância R5 comparáveis.

Família **`x2g`** de tipos de instância

Alguns casos de uso de grafos apresentam melhor desempenho quando as instâncias têm caches de grupo de buffer maiores. A família `x2g` foi lançada para oferecer maior compatibilidade com esses casos de uso. A `x2g` família tem uma taxa de memory-to-v CPU maior do que a `r6g` família `r5` or. As instâncias `x2g` também usam o processador Graviton2 e têm muitas das mesmas características de desempenho dos tipos de instância `r6g`, além de um cache maior de grupo de buffer.

Se você usa um tipo de instância `r5` ou `r6g` com baixa utilização da CPU e uma alta taxa de perda de cache do grupo de buffer, tente usar a família `x2g`. Dessa forma, você obterá a memória adicional de que precisa sem pagar por mais capacidade de CPU.

Tipo de instância **`serverless`**

O atributo Neptune Serverless pode escalar o tamanho da instância dinamicamente com base nas necessidades de recursos de uma workload. Em vez de calcular quantas vCPUs são necessárias para a aplicação, o Neptune Serverless permite que você defina limites inferior e superior na capacidade computacional (medida em unidades de capacidade do Neptune) para as instâncias no cluster de banco de dados. Workloads com utilização variável podem ser otimizadas em termos de custo usando instâncias sem servidor em vez de provisionadas.

É possível configurar instâncias provisionadas e sem servidor no mesmo cluster de banco de dados para obter uma configuração ideal de custo-desempenho.

Escolher o tipo de armazenamento certo para seu cluster de banco de dados do Neptune

O Neptune oferece dois tipos de armazenamento com um modelo de preços diferente:

- Armazenamento padrão: o armazenamento padrão fornece armazenamento econômico de banco de dados para aplicações com uso moderado a baixo de E/S.
- Armazenamento otimizado para E/S — Com o armazenamento otimizado para E/S, disponível a partir da versão 1.3.0.0 do mecanismo, você paga somente pelo armazenamento e pelas instâncias que está usando. Os custos de armazenamento são mais altos do que os do armazenamento padrão, e você não paga nada pela E/S que usa. Se seu uso de E/S for alto, o armazenamento de IOPs provisionadas pode reduzir significativamente os custos.

  O armazenamento otimizado para E/S foi desenvolvido para atender às necessidades de workloads de grafos de uso intensivo de E/S a um custo previsível, com baixa latência de E/S e throughput de E/S consistente. Você só pode alternar entre os tipos de armazenamento otimizado para E/S e padrão uma vez a cada 30 dias.

  Para obter informações sobre preços do armazenamento otimizado para E/S, consulte a página de preços do Neptune. A seção a seguir descreve como configurar o armazenamento otimizado para E/S para um cluster de banco de dados do Neptune.

Escolher o armazenamento otimizado para E/S para um cluster de banco de dados do Neptune

Por padrão, os clusters de banco de dados do Neptune usam armazenamento padrão. Você pode habilitar o armazenamento otimizado para E/S em um cluster de banco de dados no momento em que é criado, da seguinte forma:

Aqui está um exemplo de como você pode habilitar o armazenamento otimizado para E/S ao criar um cluster usando o AWS CLI:

```
aws neptune create-db-cluster \
  --database-name (name for the new database) \
  --db-cluster-identifier (an ID for the cluster) \
  --engine neptune \
  --engine-version 1.3.0.0 \
  --storage-type iopt1
```

Qualquer instância que você cria automaticamente tem o armazenamento otimizado para E/S habilitado:

```
aws neptune create-db-instance \
  --db-cluster-identifier (the ID of the new cluster) \
  --db-instance-identifier (an ID for the new instance) \
  --engine neptune \
  --db-instance-class db.r5.large
```

Você também pode modificar um cluster de banco de dados existente para habilitar o armazenamento otimizado para E/S nele, da seguinte forma:

```
aws neptune modify-db-cluster \
  --db-cluster-identifier (the ID of a cluster without I/O-Optimized storage) \
  --storage-type iopt1 \
  --apply-immediately
```

Você pode restaurar um snapshot de backup em um cluster de banco de dados com o armazenamento otimizado para E/S habilitado:

```
aws neptune restore-db-cluster-from-snapshot \
  --db-cluster-identifier (an ID for the restored cluster) \
```

```
  --snapshot-identifier (the ID of the snapshot to restore from) \
  --engine neptune \
  --engine-version 1.3.0.0 \
  --storage-type iopt1
```

Você pode determinar se um cluster está usando armazenamento otimizado para E/S com qualquer chamada `describe-`. Se o armazenamento otimizado para E/S estiver habilitado, a chamada retornará um campo de tipo de armazenamento definido como `iop1`.

Criar um cluster de banco de dados do Neptune

A maneira mais fácil de criar um novo cluster de banco de dados Amazon Neptune é usar AWS CloudFormation um modelo que cria todos os recursos necessários para você, sem precisar fazer tudo manualmente. O AWS CloudFormation modelo executa grande parte da configuração para você, incluindo a criação de uma instância do Amazon Elastic Compute Cloud (Amazon EC2):

Para iniciar um novo cluster de banco de dados Neptune usando um modelo AWS CloudFormation

1. Crie um usuário do IAM com as permissões necessárias para trabalhar com o cluster de banco de dados do Neptune, conforme explicado em Permissões de usuário do IAM.
2. Configure os pré-requisitos adicionais necessários para usar o AWS CloudFormation modelo, conforme explicado em. Pré-requisitos para usar para configurar o AWS CloudFormation Neptune
3. Invoque a AWS CloudFormation pilha, conforme descrito em. Usando uma AWS CloudFormation pilha para criar um cluster de banco de dados Neptune

Você também pode criar um banco de dados global Neptune que abrange Regiões da AWS vários, permitindo leituras globais de baixa latência e fornecendo recuperação rápida no caso raro de uma interrupção afetar um todo. Região da AWS

Para obter informações sobre a criação manual de um cluster do Amazon Neptune usando o. AWS Management ConsoleIniciar um cluster de banco de dados do Neptune usando o AWS Management Console

Você também pode usar um AWS CloudFormation modelo para criar uma função Lambda para usar com o Neptune (consulte). Usar o AWS CloudFormation para criar uma função do Lambda a ser usada no Neptune

Para obter informações gerais sobre como gerenciar clusters e instâncias no Neptune, consulte Gerenciar o banco de dados do Amazon Neptune.

Pré-requisitos para usar para configurar o AWS CloudFormation Neptune

Antes de criar um cluster do Amazon Neptune usando AWS CloudFormation um modelo, você precisa ter o seguinte:

- Um nome do par de chaves do Amazon EC2.
- As permissões necessárias para o uso AWS CloudFormation.

Crie um par de chaves do Amazon EC2 para usar na inicialização de um cluster Neptune usando AWS CloudFormation

Para iniciar um cluster de banco de dados Neptune usando AWS CloudFormation um modelo, você deve ter um par de chaves Amazon EC2 (e seu arquivo PEM associado) disponível na região em que você cria a pilha. AWS CloudFormation

Se você precisar criar o par de chaves, consulte Criar um par de chaves usando o Amazon EC2 no Guia do usuário do Amazon EC2 ou Criar um par de chaves usando o Amazon EC2 no Guia do usuário do Amazon EC2 para obter instruções.

Adicione políticas do IAM para conceder as permissões necessárias para usar o AWS CloudFormation modelo

Primeiro, você precisa ter um usuário do IAM configurado com as permissões necessárias para trabalhar com o Neptune, conforme descrito em Criar um usuário do IAM com permissões para o Neptune.

Em seguida, você precisa adicionar a política AWS gerenciada`AWSCloudFormationReadOnlyAccess`,, a esse usuário.

Por fim, você precisa criar a seguinte política gerenciada pelo cliente e adicioná-la a esse usuário:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
          "rds:CreateDBCluster",
          "rds:CreateDBInstance"
      ],
```

```
      "Resource": [
          "arn:aws:rds:*:*:*"
      ],
      "Condition": {
          "StringEquals": {
              "rds:DatabaseEngine": ["graphdb","neptune"]
          }
      }
    },
    {
      "Action": [
        "rds:AddRoleToDBCluster",
        "rds:AddSourceIdentifierToSubscription",
        "rds:AddTagsToResource",
        "rds:ApplyPendingMaintenanceAction",
        "rds:CopyDBClusterParameterGroup",
        "rds:CopyDBClusterSnapshot",
        "rds:CopyDBParameterGroup",
        "rds:CreateDBClusterParameterGroup",
        "rds:CreateDBClusterSnapshot",
        "rds:CreateDBParameterGroup",
        "rds:CreateDBSubnetGroup",
        "rds:CreateEventSubscription",
        "rds:DeleteDBCluster",
        "rds:DeleteDBClusterParameterGroup",
        "rds:DeleteDBClusterSnapshot",
        "rds:DeleteDBInstance",
        "rds:DeleteDBParameterGroup",
        "rds:DeleteDBSubnetGroup",
        "rds:DeleteEventSubscription",
        "rds:DescribeAccountAttributes",
        "rds:DescribeCertificates",
        "rds:DescribeDBClusterParameterGroups",
        "rds:DescribeDBClusterParameters",
        "rds:DescribeDBClusterSnapshotAttributes",
        "rds:DescribeDBClusterSnapshots",
        "rds:DescribeDBClusters",
        "rds:DescribeDBEngineVersions",
        "rds:DescribeDBInstances",
        "rds:DescribeDBLogFiles",
        "rds:DescribeDBParameterGroups",
        "rds:DescribeDBParameters",
        "rds:DescribeDBSecurityGroups",
        "rds:DescribeDBSubnetGroups",
```

```
        "rds:DescribeEngineDefaultClusterParameters",
        "rds:DescribeEngineDefaultParameters",
        "rds:DescribeEventCategories",
        "rds:DescribeEventSubscriptions",
        "rds:DescribeEvents",
        "rds:DescribeOptionGroups",
        "rds:DescribeOrderableDBInstanceOptions",
        "rds:DescribePendingMaintenanceActions",
        "rds:DescribeValidDBInstanceModifications",
        "rds:DownloadDBLogFilePortion",
        "rds:FailoverDBCluster",
        "rds:ListTagsForResource",
        "rds:ModifyDBCluster",
        "rds:ModifyDBClusterParameterGroup",
        "rds:ModifyDBClusterSnapshotAttribute",
        "rds:ModifyDBInstance",
        "rds:ModifyDBParameterGroup",
        "rds:ModifyDBSubnetGroup",
        "rds:ModifyEventSubscription",
        "rds:PromoteReadReplicaDBCluster",
        "rds:RebootDBInstance",
        "rds:RemoveRoleFromDBCluster",
        "rds:RemoveSourceIdentifierFromSubscription",
        "rds:RemoveTagsFromResource",
        "rds:ResetDBClusterParameterGroup",
        "rds:ResetDBParameterGroup",
        "rds:RestoreDBClusterFromSnapshot",
        "rds:RestoreDBClusterToPointInTime"
      ],
      "Effect": "Allow",
      "Resource": [
          "*"
      ]
    },
    {
      "Action": [
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:ListMetrics",
        "ec2:DescribeAccountAttributes",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcAttribute",
        "ec2:DescribeVpcs",
```

```
        "kms:ListAliases",
        "kms:ListKeyPolicies",
        "kms:ListKeys",
        "kms:ListRetirableGrants",
        "logs:DescribeLogStreams",
        "logs:GetLogEvents",
        "sns:ListSubscriptions",
        "sns:ListTopics",
        "sns:Publish"
      ],
      "Effect": "Allow",
      "Resource": [
          "*"
      ]
    },
    {
      "Action": "iam:PassRole",
      "Effect": "Allow",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
            "iam:passedToService": "rds.amazonaws.com"
        }
      }
    },
    {
      "Action": "iam:CreateServiceLinkedRole",
      "Effect": "Allow",
      "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
      "Condition": {
        "StringLike": {
            "iam:AWSServiceName": "rds.amazonaws.com"
        }
      }
    }
  ]
}
```

> **Note**
>
> As seguintes permissões são necessárias apenas para excluir uma pilha: `iam:DeleteRole`, `iam:RemoveRoleFromInstanceProfile`, `iam:DeleteRolePolicy`, `iam:DeleteInstanceProfile` e `ec2:DeleteVpcEndpoints`.
> Observe também que `ec2:*Vpc` concede `ec2:DeleteVpc` permissões.

Usando uma AWS CloudFormation pilha para criar um cluster de banco de dados Neptune

Você pode usar um AWS CloudFormation modelo para configurar um cluster de banco de dados Neptune.

1. Para iniciar a AWS CloudFormation pilha no AWS CloudFormation console, escolha um dos botões Iniciar pilha na tabela a seguir.

Região	Visualização	Visualizar no Designer	Executar
Leste dos EUA (Norte da Virgínia)	Visão	Visualizar no Designer	
Leste dos EUA (Ohio)	Visão	Visualizar no Designer	
Oeste dos EUA (N. da Califórnia)	Visão	Visualizar no Designer	
Oeste dos EUA (Oregon)	Visão	Visualizar no Designer	
Canadá (Central)	Visão	Visualizar no Designer	
América do Sul (São Paulo)	Visão	Visualizar no Designer	
Europa (Estocolmo)	Visão	Visualizar no Designer	
Europa (Irlanda)	Visão	Visualizar no Designer	
Europa (Londres)	Visão	Visualizar no Designer	

Região	Visualização	Visualizar no Designer	Executar
Europa (Paris)	Visão	Visualizar no Designer	
Europa (Frankfurt)	Visão	Visualizar no Designer	
Oriente Médio (Barém)	Visão	Visualizar no Designer	
Oriente Médio (Emirados Árabes Unidos)	Visão	Visualizar no Designer	
Israel (Tel Aviv)	Visão	Visualizar no Designer	
África (Cidade do Cabo)	Visão	Visualizar no Designer	
Ásia-Pacífico (Hong Kong)	Visão	Visualizar no Designer	
Ásia-Pacífico (Tóquio)	Visão	Visualizar no Designer	
Ásia-Pacífico (Seul)	Visão	Visualizar no Designer	
Ásia-Pacífico (Singapura)	Visão	Visualizar no Designer	
Ásia-Pacífico (Sydney)	Visão	Visualizar no Designer	
Ásia-Pacífico (Mumbai)	Visão	Visualizar no Designer	

Região	Visualização	Visualizar no Designer	Executar
China (Pequim)	Visão	Visualizar no Designer	Launch Stack
China (Ningxia)	Visão	Visualizar no Designer	Launch Stack
AWS GovCloud (Oeste dos EUA)	Visão	Visualizar no Designer	Launch Stack
AWS GovCloud (Leste dos EUA)	Visão	Visualizar no Designer	Launch Stack

2. Na página Select Template, escolha Next.
3. Na página Especificar detalhes, escolha um par de chaves para o nome EC2SSH KeyPair.

   Esse par de chaves é necessário para acessar a instância do EC2. Verifique se você tem o arquivo PEM para o par de chaves selecionado.
4. Escolha Próximo.
5. Na página Options (Opções), escolha Next (Avançar).
6. Na página Revisar, marque a primeira caixa de seleção para confirmar que o AWS CloudFormation criará recursos do IAM. Marque a segunda caixa de seleção para confirmar `CAPABILITY_AUTO_EXPAND` para a nova pilha.

   > **Note**
   >
   > `CAPABILITY_AUTO_EXPAND` confirma explicitamente que os macros serão expandidos ao criar a pilha, sem revisão anterior. Os usuários geralmente criam um conjunto de alterações a partir de um modelo processado para que as alterações feitas pelos macros possam ser revisadas antes de criar a pilha. Para obter mais informações, consulte a API AWS CloudFormation CreateStack.

   Em seguida, selecione Criar.

> **Note**
>
> Você também pode usar seu AWS CloudFormation modelo para atualizar a versão do mecanismo do seu cluster de banco de dados.

Configurar a Amazon VPC onde o cluster de banco de dados do Amazon Neptune está localizado

Um cluster de banco de dados do Amazon Neptune só pode ser criado em uma Amazon Virtual Private Cloud (Amazon VPC). Seus endpoints são acessíveis dentro dessa VPC.

Há várias maneiras diferentes de configurar a VPC, dependendo de como você deseja acessar o cluster de banco de dados.

Aqui estão alguns fatores que você deve ter em mente ao configurar a VPC em que o cluster de banco de dados do Neptune está localizado:

- A VPC deve ter pelo menos duas sub-redes. As duas sub-redes devem estar em diferentes zonas de disponibilidade (AZs). Ao distribuir as instâncias de cluster em pelo menos duas zonas de disponibilidade, você ajuda a garantir que sempre haverá instâncias disponíveis no cluster de banco de dados no caso improvável de ocorrer uma falha na zona de disponibilidade. O volume do cluster de banco de dados do Neptune sempre abrange três AZs para fornecer armazenamento durável com probabilidade extremamente baixa de perda de dados.
- Os blocos CIDR em cada sub-rede devem ser grandes o suficiente para fornecer endereços IP de que o Neptune possa precisar durante atividades de manutenção, failover e escalabilidade.
- A VPC deve ter um grupo de sub-redes de banco de dados que contenha sub-redes criadas. O Neptune seleciona uma das opções do grupo de sub-redes e um endereço IP dentro dessa sub-rede para associar a cada instância de banco de dados no cluster de banco de dados. A instância de banco de dados fica então localizada na mesma AZ da sub-rede.
- A VPC deve ter o DNS ativado (nomes de host DNS e resolução de DNS).
- A VPC deve ter um grupo de segurança da VPC que permita o acesso ao cluster de banco de dados.
- A locação em uma VPC do Neptune deve ser definida como Padrão.

Adicionar sub-redes à VPC em que o cluster de banco de dados do Neptune está localizado

Uma sub-rede é um intervalo de endereços IP na VPC. É possível iniciar recursos, como um cluster de banco de dados do Neptune ou uma instância do EC2 em uma sub-rede específica. Ao criar uma sub-rede, você especifica o bloco CIDR IPv4 para a sub-rede, o qual é um subconjunto do bloco CIDR da VPC. Cada sub-rede deve residir inteiramente dentro de uma zona de disponibilidade (AZ) e não pode abranger zonas. Ao iniciar as instâncias em zonas de disponibilidade separadas, você pode proteger suas aplicações contra falhas de uma das zonas. Para obter mais informações, consulte a documentação de sub-redes da VPC.

Um cluster de banco de dados do Neptune exige pelo menos duas sub-redes da VPC.

Como adicionar sub-redes a uma VPC

1. Faça login AWS Management Console e abra o console da Amazon VPC em https://console.aws.amazon.com/vpc/.
2. No painel de navegação, escolha Sub-redes.
3. No Painel da VPC, selecione Sub-redes e, depois, escolha Criar sub-rede.
4. Na página Criar sub-rede, selecione a VPC na qual você deseja criar a sub-rede.
5. Em Configurações de sub-rede, faça as seguintes seleções:
    a. Insira um nome para a nova sub-rede em Nome da sub-rede.
    b. Escolha uma zona de disponibilidade (AZ) para a sub-rede ou deixe a opção em Sem preferências.
    c. Insira o bloco de endereços IP da sub-rede no Bloco CIDR IPv4.
    d. Adicione tags à sub-rede, se necessário.
    e. Escolher
6. Se você quiser criar outra sub-rede ao mesmo tempo, escolha Adicionar nova sub-rede.
7. Selecione Criar sub-rede para criar as sub-redes.

Criar um grupo de sub-redes na VPC

Crie um grupo de sub-redes.

Como criar um grupo de sub-redes do Neptune

1. Faça login no AWS Management Console e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. Escolha Subnet groups (Grupos de sub-redes) e Create DB Subnet Group (Criar grupo de sub-redes de banco de dados).
3. Insira um nome e uma descrição para o grupo de sub-redes (a descrição é obrigatória).
4. Em VPC, escolha a VPC em que você deseja que esse grupo de sub-redes esteja localizado.
5. Em Zona de disponibilidade, escolha a AZ em que você deseja que esse grupo de sub-redes esteja localizado.
6. Em Sub-rede, adicione uma ou mais das sub-redes dessa AZ a esse grupo de sub-redes.
7. Selecione Criar para criar o grupo de sub-redes.

Criar um grupo de segurança usando o console da VPC

Grupos de segurança concedem acesso ao cluster de banco de dados do Neptune na VPC. Eles atuam como um firewall para o cluster de banco de dados associado controlando o tráfego de entrada e de saída em nível de instância. Por padrão, uma instância de banco de dados é criada com um firewall e um grupo de segurança padrão que impedem o acesso a ela. Para habilitar o acesso, é necessário ter um grupo de segurança da VPC com regras adicionais.

O procedimento a seguir mostra como adicionar uma regra de TCP personalizada que especifique o intervalo de portas e endereços IP que a instância do Amazon EC2 usa para acessar o cluster de banco de dados do Neptune. É possível usar o grupo de segurança da VPC atribuído à instância do EC2 em vez do endereço IP.

Como criar um grupo de segurança da VPC para o Neptune no console

1. Faça login AWS Management Console e abra o console da Amazon VPC em https://console.aws.amazon.com/vpc/.
2. No canto superior direito do console, escolha a AWS região em que você deseja criar um grupo de segurança de VPC para o Neptune. A lista de recursos da Amazon VPC para essa região deve indicar que você tem pelo menos uma VPC e várias sub-redes. Se isso não acontecer, você não terá uma VPC padrão nessa região.
3. No painel de navegação, em Segurança, escolha Grupos de segurança.

4. Escolha Create security group (Criar grupo de segurança). Na janela Criar grupo de segurança, insira o Nome do grupo de segurança, uma Descrição e o identificador da VPC em que o cluster de banco de dados do Neptune residirá.
5. Adicione uma regra de entrada para o grupo de segurança de uma instância do Amazon EC2 que você deseja conectar ao cluster de banco de dados do Neptune:
   a. Na área Regras de entrada, selecione Adicionar regra.
   b. Na lista Tipo, deixe TCP personalizado selecionado.
   c. Na caixa Intervalo de portas, digite 8182, o valor da porta padrão para o Neptune.
   d. Em Origem, insira o intervalo de endereços IP (valor CIDR) a partir do qual você acessará o Neptune ou selecione um nome de grupo de segurança existente.
   e. Se for necessário adicionar mais endereços IP ou intervalos de portas diferentes, selecione Adicionar regra novamente.
6. Na área Regras de saída, você também pode adicionar uma ou mais regras de saída, se necessário.
7. Quando terminar, escolha Create security group (Criar grupo de segurança).

Você pode usar esse grupo de segurança da VPC ao criar um cluster de banco de dados do Neptune.

Se usar uma VPC padrão, já será criado para você um grupo de sub-redes padrão distribuídas por todas as sub-redes da VPC. Ao escolher Criar banco de dados no console do Neptune, a VPC padrão é usada, a menos que você especifique uma diferente.

Verificar se há compatibilidade com DNS na VPC

Domain Name System (DNS) é um padrão por meio do qual os nomes usados na Internet são determinados de acordo com os endereços IP correspondentes. Um nome de host do DNS denomina exclusivamente um computador e consiste em um nome de host e em um nome de domínio. Os servidores DNS determinam os nomes do host DNS de acordo com os endereços IP correspondentes.

Certifique-se de que os nomes de host e a resolução do DNS estejam habilitados na VPC. Os atributos de rede `enableDnsHostnames` e `enableDnsSupport` da VPC precisam ser definidos como `true`. Para visualizar e modificar esses atributos, acesse o console da VPC em https://console.aws.amazon.com/vpc/.

Para obter mais informações, consulte Como usar o DNS com sua VPC.

> **Note**
>
> Se você estiver usando o Route 53, confirme que a configuração não substitui os atributos de rede do DNS na VPC.

Conectar-se ao grafo do Amazon Neptune

Depois de criar um cluster de banco de dados do Neptune, a próxima etapa é configurar a conexão com ele.

Configurar o `curl` ou o awscurl para comunicar-se com o endpoint do Neptune

Ter uma ferramenta de linha de comando para enviar consultas ao cluster de banco de dados do Neptune é muito útil, conforme ilustrado em muitos dos exemplos desta documentação. A ferramenta de linha de comando curl é uma excelente opção para se comunicar com os endpoints do Neptune quando a autenticação do IAM não está ativada. As versões que começam com 7.75.0 são compatíveis com a opção `--aws-sigv4` para assinar solicitações quando a autenticação do IAM está ativada.

Para endpoints em que a autenticação do IAM está ativada, você também pode usar o awscurl, que usa quase exatamente a mesma sintaxe, mas o `curl` é compatível com solicitações de assinatura conforme necessário para a autenticação do IAM. Devido à segurança adicional oferecida pela =autenticação do IAM, geralmente é uma boa ideia habilitá-la.

Para obter informações sobre como usar o `curl` (ou o `awscurl`), consulte a curl man page e o guia Everything curl.

Para se conectar usando HTTPS (exigido pelo Neptune), o `curl` precisa acessar os certificados apropriados. Desde que o `curl` possa localizar os certificados adequados, ele trata as conexões HTTPS da mesma forma como as conexões HTTP, sem parâmetros extras. O mesmo se aplica ao `awscurl`. Os exemplos desta documentação se baseiam nesse cenário.

Para saber como obter esses certificados e como formatá-los corretamente em um armazenamento de certificados (CA) que o `curl` possa usar, consulte SSL Certificate Verification na documentação do `curl`.

Depois, você pode especificar o local desse armazenamento de certificados CA usando a variável de ambiente `CURL_CA_BUNDLE`. No Windows, o `curl` os procura automaticamente em um arquivo chamado `curl-ca-bundle.crt`. Ele procura primeiro no mesmo diretório `curl.exe` e, em seguida, em outros lugares no caminho. Para obter mais informações, consulte SSL Certificate Verification.

Diferentes maneiras de se conectar a um cluster de banco de dados do Neptune

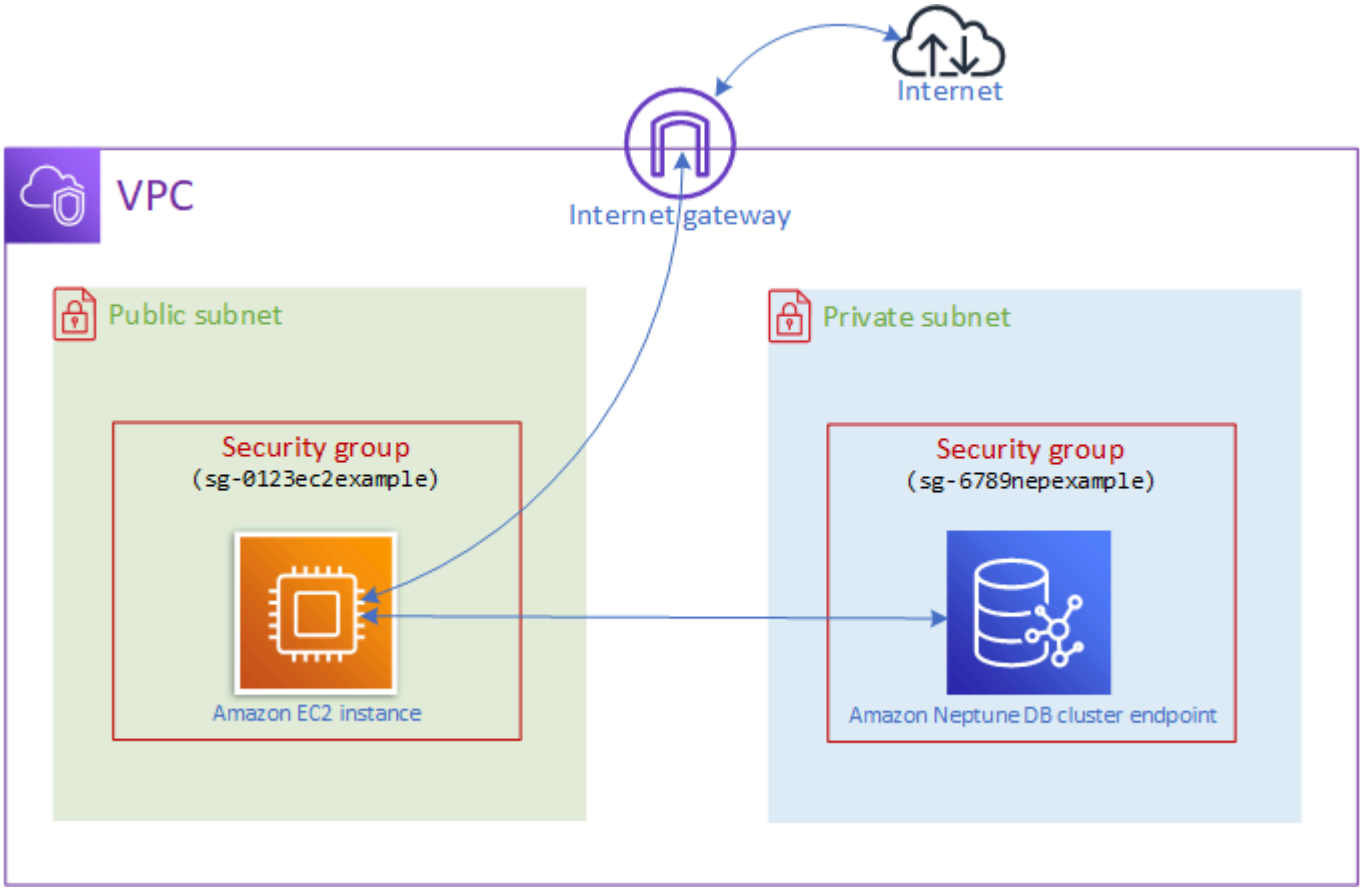
Um cluster de banco de dados do Amazon Neptune só pode ser criado em uma Amazon Virtual Private Cloud (Amazon VPC). A menos que você habilite e configure os endpoints públicos do Neptune para o cluster de banco de dados, seus endpoints serão acessíveis somente dentro dessa VPC.

Há várias maneiras diferentes de configurar o acesso ao cluster de banco de dados do Neptune na VPC:

- Conectar-se por uma instância do Amazon EC2 na mesma VPC.
- Conectar-se por uma instância do Amazon EC2 na mesma VPC.
- Conectar-se por uma rede privada.

Conectar-se a um cluster de banco de dados do Neptune por uma instância do Amazon EC2 na mesma VPC.

Uma das maneiras mais comuns de se conectar a um banco de dados do Neptune é usando uma instância do Amazon EC2 na mesma VPC do cluster de banco de dados do Neptune. Por exemplo, a instância do EC2 pode estar executando um servidor web que se conecte à Internet. Nesse caso, somente a instância do EC2 tem acesso ao cluster de banco de dados do Neptune, e a Internet só tem acesso à instância do EC2:



Para habilitar essa configuração, você precisa ter os grupos de segurança e os grupos de sub-rede corretos da VPC configurados. O servidor web está hospedado em uma sub-rede pública, para que ele possa acessar a Internet pública, e a instância de cluster do Neptune é hospedada em uma sub-rede privada para mantê-la segura. Consulte Configurar a Amazon VPC onde o cluster de banco de dados do Amazon Neptune está localizado.

Para que a instância do Amazon EC2 se conecte ao endpoint do Neptune, por exemplo, na porta 8182, será necessário configurar um grupo de segurança para fazer isso. Se a instância do Amazon EC2 estiver usando um grupo de segurança denominado, por exemplo, `ec2-sg1`, será necessário criar outro grupo de segurança do Amazon EC2 (digamos `db-sg1`) com regras de entrada para a porta 8182 com `ec2-sg1` como a fonte. Depois, adicione `db-sg1` ao cluster do Neptune para viabilizar a conexão.

Depois de criar a instância do Amazon EC2, você pode fazer login nela usando SSH e se conectar ao cluster de banco de dados do Neptune. Para obter informações sobre como se conectar a uma instância do EC2 usando SSH, consulte Conecte-se à sua instância Linux no Guia do usuário do Amazon EC2.

Se você estiver usando uma linha de comando Linux ou macOS para se conectar à instância do EC2, poderá colar o comando SSH do item SSHAccess na seção Saídas da pilha. AWS CloudFormation Você deve ter o arquivo PEM no diretório atual e as permissões do arquivo PEM devem ser definidas como 400 (`chmod 400 keypair.pem`).

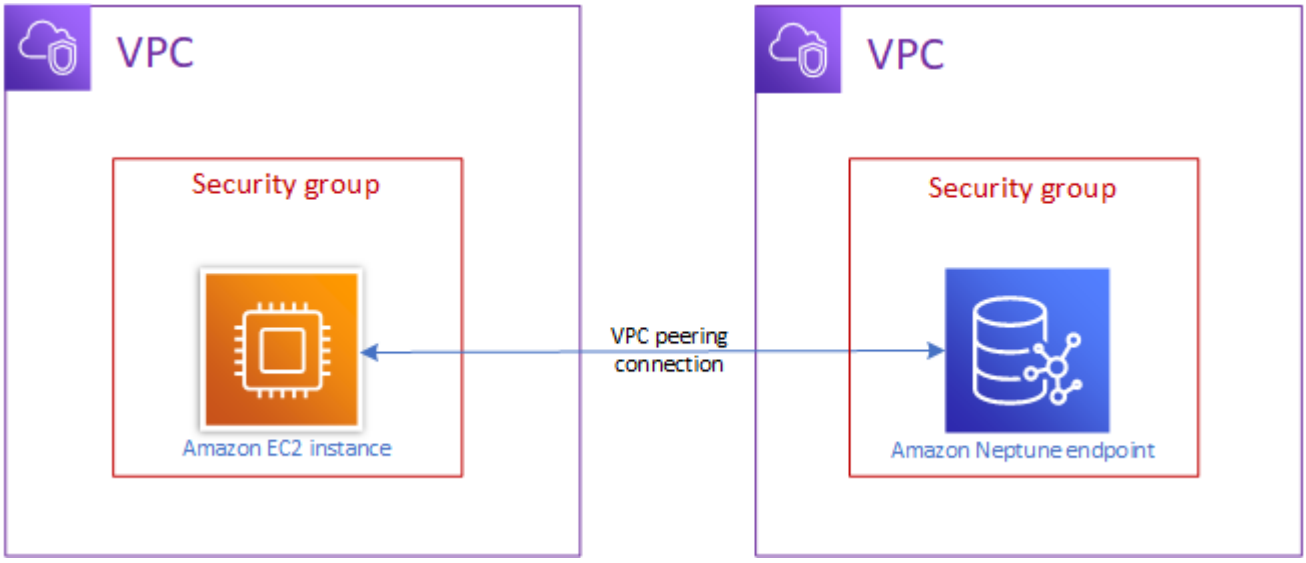
Como criar uma VPC com sub-redes públicas e privadas

1. Faça login AWS Management Console e abra o console da Amazon VPC em https://console.aws.amazon.com/vpc/.
2. No canto superior direito do AWS Management Console, escolha a região na qual criar sua VPC.
3. No Painel da VPC, selecione Iniciar assistente da VPC.
4. Preencha a área Configurações da VPC da página Criar VPC:
    a. Em Resources to create (Recursos a serem criados), escolha VPC, subnets, etc. (VPC, sub-redes etc.).
    b. Deixe a tag de nome padrão como está, digite um nome de sua escolha ou desmarque a caixa de seleção Gerar automaticamente para desativar a geração da tags de nome.
    c. Deixe o valor do bloco CIDR IPv4 em `10.0.0.0/16`.
    d. Deixe o valor do bloco CIDR IPv6 em Nenhum bloco CIDR.
    e. Deixe a Locação como Padrão.
    f. Deixe o número de Zonas de disponibilidade (AZs) em 2.
    g. Deixe os Gateways NAT ($) em Nenhum, a menos que você precise de um ou mais gateways NAT.
    h. Defina Endpoints da VPC como Nenhum, a menos que você use o Amazon S3.
    i. As opções Habilitar nomes de host do DNS e Habilitar resolução DNS devem estar selecionadas.
5. Escolha Criar VPC.

Acessar o cluster de banco de dados por uma instância do Amazon EC2 em outra VPC

Um cluster de banco de dados do Amazon Neptune só pode ser criado em uma Amazon Virtual Private Cloud (Amazon VPC) e seus endpoints só são acessíveis dentro dessa VPC, em geral, por uma instância do Amazon Elastic Compute Cloud (Amazon EC2) em execução nessa VPC.

Quando o cluster de banco de dados estiver em uma VPC diferente da instância do EC2 que você está usando para acessá-lo, você pode usar o emparelhamento da VPC para estabelecer a conexão:



Conexão de emparelhamento da VPC é uma conexão de rede entre duas VPCs que direciona o tráfego entre elas de forma privada, para que as instâncias em qualquer uma das VPCs possam se comunicar como se estivessem na mesma rede. Você pode criar uma conexão de emparelhamento de VPC entre VPCs na sua conta, entre uma VPC na sua conta AWS e uma VPC em outra conta ou com uma VPC em uma AWS região diferente. AWS

AWS usa a infraestrutura existente de uma VPC para criar uma conexão de emparelhamento de VPC. Não é um gateway nem uma conexão AWS VPN Site-to-Site e não depende de um hardware físico separado. Não há um ponto único de falha de comunicação nem um gargalo de largura de banda.

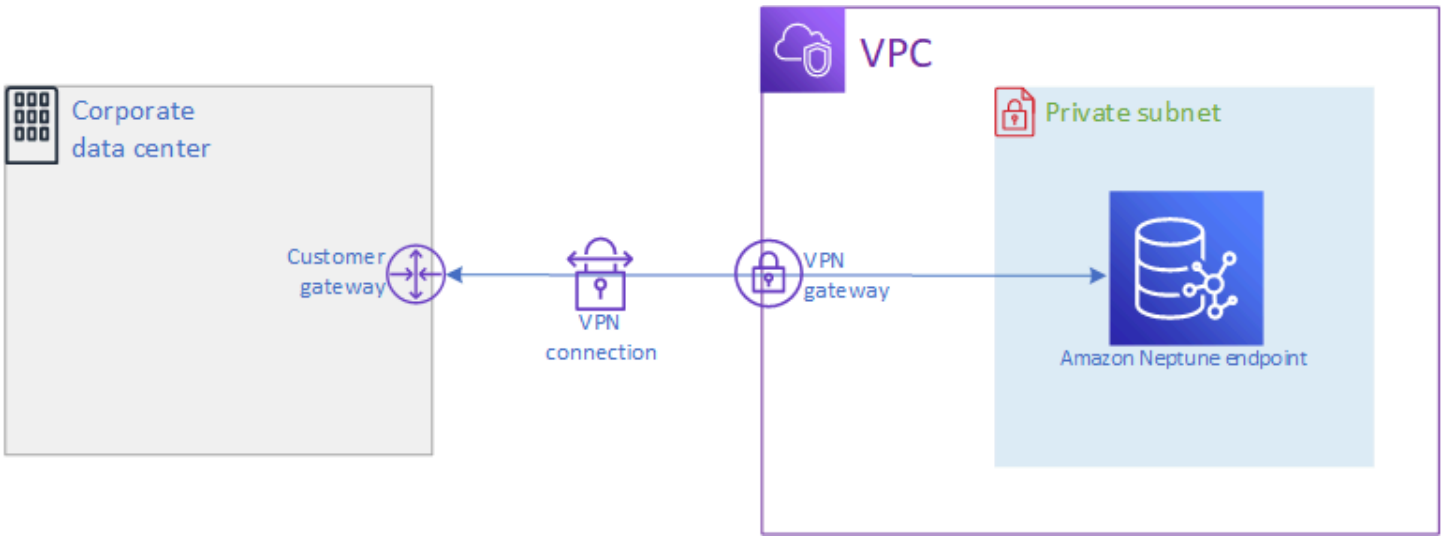
Consulte o Guia de emparelhamento da Amazon VPC para obter mais informações sobre como usar o emparelhamento da VPC.

Acessar o cluster de banco de dados por uma rede privada

Você pode acessar um cluster de banco de dados do Neptune por uma rede privada de duas maneiras diferentes:

- Usando uma conexão AWS Site-to-Site VPN.
- Usando uma conexão AWS Direct Connect.

Os links acima têm informações sobre esses métodos de conexão e como configurá-los. A configuração de uma conexão AWS site a site pode ter a seguinte aparência:



Proteger os dados no Amazon Neptune

Há várias maneiras de proteger os clusters do Amazon Neptune.

Usar políticas do IAM para restringir o acesso a um cluster de banco de dados do Neptune

Para controlar quem pode realizar ações de gerenciamento do Neptune nos clusters e instâncias de banco de dados do Neptune, use (IAM). AWS Identity and Access Management

Ao usar uma conta do IAM para acessar o console do Neptune, você deve primeiro fazer login usando sua conta do IAM antes de abrir AWS Management Console o console do Neptune em https://console.aws.amazon.com/neptune/home.

Quando você se conecta AWS usando as credenciais do IAM, sua conta do IAM deve ter políticas do IAM que concedam as permissões necessárias para realizar as operações de gerenciamento do Neptune. Para ter mais informações, consulte Usar diferentes tipos de política do IAM para controle de acesso ao Neptune.

Usar grupos de segurança da VPC para restringir o acesso a um cluster de banco de dados do Neptune

Os clusters de banco de dados do Neptune devem ser criados em uma Amazon Virtual Private Cloud (VPC). Para controlar quais dispositivos e instâncias do EC2 podem abrir conexões com o endpoint e

a porta da instância de banco de dados para clusters de banco de dados do Neptune em uma VPC, use um grupo de segurança da VPC. Para obter mais informações sobre VPCs, consulte Criar um grupo de segurança usando o console da VPC.

Usar a autenticação do IAM para restringir o acesso a um cluster de banco de dados do Neptune

Se você habilitar a autenticação AWS Identity and Access Management (IAM) em um cluster de banco de dados Neptune, qualquer pessoa que acesse o cluster de banco de dados deverá primeiro ser autenticada. Para obter informações sobre a configuração da autenticação do IAM, consulte Visão geral do AWS Identity and Access Management (IAM) no Amazon Neptune.

Para obter informações sobre o uso de credenciais temporárias para autenticação, incluindo exemplos para AWS CLI, AWS Lambda, e Amazon EC2, consulte. the section called “Credenciais temporárias”

Os seguintes links fornecem informações adicionais sobre como se conectar ao Neptune usando a autenticação do IAM com as linguagens de consulta individuais:

Usar o Gremlin com autenticação do IAM

- the section called “Console do Gremlin”
- the section called “Gremlin Java”
- the section called “Exemplo Python”

> **Note**
>
> Este exemplo se aplica ao Gremlin e SPARQL.

Usar o openCypher com autenticação do IAM

- the section called “Console do Gremlin”
- the section called “Gremlin Java”
- the section called “Exemplo Python”

> **Note**
>
> Este exemplo se aplica ao Gremlin e SPARQL.

Usar o SPARQL com autenticação do IAM

- the section called “SPARQL Java (RDF4J e Jena)”
- the section called “Exemplo Python”

> **Note**
>
> Este exemplo se aplica ao Gremlin e SPARQL.

Conceitos básicos sobre o acesso ao grafo do Neptune

Depois de criar um cluster de banco de dados do Neptune e configurar a conexão com ele, a próxima etapa é se comunicar com ele para carregar dados, fazer consultas, etc. Para fazer isso, a maioria das pessoas usa as ferramentas de linha de comando `curl` ou `awscurl`.

Configurar o `curl` para comunicar-se com o endpoint do Neptune

Conforme ilustrado em muitos exemplos desta documentação, a ferramenta de linha de comando curl é uma opção útil para comunicação com o endpoint do Neptune. Para obter informações sobre a ferramenta, consulte a curl man page e o guia Everything curl.

Para conectar-se usando HTTPS (conforme recomendamos e como o Neptune exige na maioria das regiões), o `curl` precisa acessar os certificados adequados. Para saber como obter esses certificados e como formatá-los corretamente em um armazenamento de certificados CA que o `curl` pode usar, consulte SSL Certificate Verification na documentação do `curl`.

Depois, você pode especificar o local desse armazenamento de certificados CA usando a variável de ambiente `CURL_CA_BUNDLE`. No Windows, o `curl` os procura automaticamente em um arquivo chamado `curl-ca-bundle.crt`. Ele procura primeiro no mesmo diretório `curl.exe` e, em seguida, em outros lugares no caminho. Para obter mais informações, consulte SSL Certificate Verification.

Desde que o `curl` possa localizar os certificados adequados, ele trata as conexões HTTPS da mesma forma como as conexões HTTP, sem parâmetros extras. Os exemplos desta documentação se baseiam nesse cenário.

Usar uma linguagem de consulta para acessar dados do grafo no cluster de banco de dados do Neptune

Depois de se conectar, você pode usar as linguagens de consulta Gremlin e openCypher para criar e consultar um grafo de propriedades, ou a linguagem de consulta SPARQL para criar e consultar um grafo contendo dados RDF.

Linguagens de consulta de grafo compatíveis com o Neptune

- O Gremlin é uma linguagem de percurso de grafos de propriedades. Uma consulta no Gremlin é um percurso composto por etapas distintas, cada uma das quais segue uma borda até um nó. Consulte a documentação do Gremlin no Apache TinkerPop 3 para obter mais informações.

  A implementação do Gremlin no Neptune tem algumas diferenças em relação a outras implementações, principalmente ao usar o Gremlin-Groovy (as consultas do Gremlin enviadas como texto serializado). Para ter mais informações, consulte Conformidade com os padrões do Gremlin no Amazon Neptune.
- openCypher é uma linguagem de consulta declarativa para grafos de propriedades originalmente desenvolvida pela Neo4j, que se tornou de código aberto em 2015, e contribuiu para o projeto openCypher sob uma licença de código aberto Apache 2. A sintaxe dela está documentada na Cypher Query Language Reference, versão 9.
- O SPARQL é uma linguagem de consulta declarativa para dados do RDF baseada na correspondência do padrão de grafo padronizado pelo World Wide Web Consortium (W3C) e descrito em SPARQL 1.1 Overview e na especificação SPARQL 1.1 Query Language.

> **Note**
>
> É possível acessar os dados do grafo de propriedades em Neptune usando o Gremlin e o openCypher, mas não com o SPARQL. Da mesma forma, você só pode acessar dados RDF usando SPARQL, não o Gremlin nem o openCypher.

Usar o Gremlin para acessar o grafo no Amazon Neptune

Você pode usar o Gremlin Console para experimentar TinkerPop gráficos e consultas em um ambiente REPL (loop). read-eval-print

O tutorial a seguir descreve o uso do console do Gremlin para adicionar propriedades, vértices, bordas e muito mais a um grafo do Neptune e destaca algumas diferenças na implementação do Gremlin específica do Neptune.

> **Note**
>
> Este exemplo pressupõe que você tenha concluído o seguinte:
>
> - Você se conectou usando SSH a uma instância do Amazon EC2.
> - Você criou um cluster do Neptune conforme detalhado em Criar um cluster de banco de dados.
> - Você instalou o console do Gremlin conforme descrito em Instalar o console do Gremlin.

Usar o Gremlin Console

1. Altere os diretórios para a pasta na qual os arquivos do Gremlin Console são descompactados.

   ```
   cd apache-tinkerpop-gremlin-console-3.6.5
   ```

2. Digite o comando a seguir para executar o Gremlin Console.

   ```
   bin/gremlin.sh
   ```

   A seguinte saída deverá ser mostrada:

   ```
           \,,,/
           (o o)
   -----oOOo-(3)-oOOo-----
   plugin activated: tinkerpop.server
   plugin activated: tinkerpop.utilities
   plugin activated: tinkerpop.tinkergraph
   gremlin>
   ```

Agora você está no prompt do `gremlin>`. Você insere as etapas restantes nesse prompt.

3. No prompt `gremlin>`, insira o seguinte para conectar-se a instâncias de banco de dados do Neptune.

```
:remote connect tinkerpop.server conf/neptune-remote.yaml
```

4. No prompt `gremlin>`, insira o seguinte para alternar para modo remoto. Isso envia todas as consultas do Gremlin para a conexão remota.

```
:remote console
```

5. Adicionar vértice com rótulo e propriedade.

```
g.addV('person').property('name', 'justin')
```

É atribuído um ID de `string` ao vértice que contém um GUID. Todos os IDs de vértice são strings no Neptune.

6. Adicionar um vértice com id personalizado.

```
g.addV('person').property(id, '1').property('name', 'martin')
```

A propriedade `id` não é mencionada. Ela é uma palavra-chave para o ID do vértice. O ID do vértice aqui é uma string com o número 1 nela.

Os nomes de propriedade normais devem estar entre aspas.

7. Alterar ou adicionar propriedade caso não exista.

```
g.V('1').property(single, 'name', 'marko')
```

Aqui você está alterando a propriedade `name` para o vértice da etapa anterior. Isso remove todos os valores existentes da propriedade `name`.

Se você não especificar `single`, em vez disso, ele acrescenta o valor à propriedade `name` caso ainda não tenha feito.

8. Adicionar propriedade, mas incluir propriedade caso já possua um valor.

```
g.V('1').property('age', 29)
```

O Neptune usa cardinalidade set como a ação padrão.

Esse comando adiciona a propriedade `age` com o valor 29, mas não substitui qualquer valor existente.

Se a propriedade `age` já tinha um valor, esse comando anexa 29 à propriedade. Por exemplo, se a propriedade `age` era 27, o novo valor será `[ 27, 29 ]`.

9. Adicionar vários vértices.

```
g.addV('person').property(id, '2').property('name', 'vadas').property('age',
 27).iterate()
g.addV('software').property(id, '3').property('name', 'lop').property('lang',
 'java').iterate()
g.addV('person').property(id, '4').property('name', 'josh').property('age',
 32).iterate()
g.addV('software').property(id, '5').property('name', 'ripple').property('lang',
 'java').iterate()
g.addV('person').property(id, '6').property('name', 'peter').property('age', 35)
```

Você pode enviar várias declarações ao mesmo tempo ao Neptune.

As instruções podem ser separadas por novas linhas (`'\n'`), espaços (`' '`), ponto e vírgula (`';'`) ou nada (por exemplo: `g.addV('person').iterate()g.V()` é válido).

> **Note**
>
> O Gremlin Console envia um comando separado a cada nova linha (`'\n'`), de modo que cada um será uma transação separada nesse caso. Esse exemplo tem todos os comandos em linhas separadas para legibilidade. Remova os caracteres de nova linha (`'\n'`) para enviá-lo como um único comando por meio do Gremlin Console.

Todas as instruções diferentes da última instrução devem terminar em uma etapa de encerramento, como `.next()` ou `.iterate()`, ou não serão executadas. O Gremlin Console não requer essas etapas de encerramento. Use `.iterate` sempre que não precisar que os resultados sejam serializados.

Todas as instruções enviadas em conjunto são incluídas em uma única transação e são bem-sucedidas ou falham em conjunto.

10. Adicionar bordas.

```
g.V('1').addE('knows').to(__.V('2')).property('weight', 0.5).iterate()
g.addE('knows').from(__.V('1')).to(__.V('4')).property('weight', 1.0)
```

Veja aqui duas maneiras diferentes para adicionar uma borda.

11. Adicionar o restante do gráfico moderno.

```
g.V('1').addE('created').to(__.V('3')).property('weight', 0.4).iterate()
g.V('4').addE('created').to(__.V('5')).property('weight', 1.0).iterate()
g.V('4').addE('knows').to(__.V('3')).property('weight', 0.4).iterate()
g.V('6').addE('created').to(__.V('3')).property('weight', 0.2)
```

12. Excluir um vértice.

```
g.V().has('name', 'justin').drop()
```

Remove o vértice com a propriedade `name` igual a `justin`.

> **Important**
>
> Pare aqui e você terá o gráfico completo do Apache TinkerPop Modern. Os exemplos na seção Traversal da TinkerPop documentação usam o gráfico moderno.

13. Executar um percurso.

```
g.V().hasLabel('person')
```

Retorna todos os vértices `person`.

14. Executar um percurso com valores (valueMap()).

```
g.V().has('name', 'marko').out('knows').valueMap()
```

Retorna pares de chave-valor para todos os vértices que `marko` "conhece".

15. Especificar vários rótulos.

```
g.addV("Label1::Label2::Label3")
```

O Neptune é compatível com vários rótulos para um vértice. Quando cria um rótulo, você pode especificar vários rótulos separados com `::`.

Esse exemplo adiciona um vértice com três diferentes rótulos.

A etapa `hasLabel` corresponde esse vértice com qualquer um destes três rótulos: `hasLabel("Label1")`, `hasLabel("Label2")` e `hasLabel("Label3")`.

O delimitador `::` é reservado somente para esse uso.

Você não pode especificar vários rótulos na etapa `hasLabel`. Por exemplo, `hasLabel("Label1::Label2")` não corresponde a nada.

16. Especifique hora/data.

```
g.V().property(single, 'lastUpdate', datetime('2018-01-01T00:00:00'))
```

O Neptune não é compatível com Java Date. Use a função `datetime()`. O `datetime()` aceita uma string `datetime` compatível com ISO8061.

Compatível com os formatos: `YYYY-MM-DD`, `YYYY-MM-DDTHH:mm`, `YYYY-MM-DDTHH:mm:SS` e `YYYY-MM-DDTHH:mm:SSZ`.

17. Excluir vértices, propriedades ou bordas.

```
g.V().hasLabel('person').properties('age').drop().iterate()
g.V('1').drop().iterate()
g.V().outE().hasLabel('created').drop()
```

Aqui estão alguns exemplos de remoção.

> **Note**
>
> A etapa `.next()` não funciona com `.drop()`. Use `.iterate()` em vez disso.

18. Ao terminar, insira o seguinte para sair do Gremlin Console.

```
:exit
```

> **Note**
>
> Use um ponto e vírgula (`;`) ou um caractere de nova linha (`\n`) para separar cada instrução. Cada travessia anterior à travessia final deve terminar em `iterate()` a ser executada. Somente os dados da travessia final são retornados.

Usar o openCypher para acessar o grafo no Amazon Neptune

Para começar a usar o OpenCypher, consulte ou use os cadernos openCypher OpenCypher no repositório de cadernos gráficos Neptune. GitHub

Usar o RDF e o SPARQL para acessar o grafo no Amazon Neptune

O SPARQL é uma linguagem de consulta para o Resource Description Framework (RDF), que é um formato de dados de grafos projetado para a web. O Amazon Neptune é compatível com o SPARQL 1.1. Isso significa que você pode se conectar a uma instância de banco de dados do Neptune e consultar o grafo usando a linguagem de consulta descrita na especificação SPARQL 1.1 Query Language.

Uma consulta no SPARQL consiste em uma cláusula `SELECT` para especificar as variáveis a serem retornadas e uma cláusula `WHERE` para especificar quais dados corresponder no gráfico. Se não estiver familiarizado com as consultas do SPARQL, consulte Writing Simple Queries no SPARQL 1.1 Query Language.

O endpoint HTTP para consultas do SPARQL a uma instância de banco de dados do Neptune é `https://`*`your-neptune-endpoint`*`:`*`port`*`/sparql`.

Para se conectar ao SPARQL

1. Você pode obter o endpoint SPARQL para seu cluster Neptune a partir do SparqlEndpointitem na seção Saídas da pilha. AWS CloudFormation
2. Digite o seguinte para enviar um SPARQL **`UPDATE`** usando HTTP `POST` e o comando curl.

```
curl -X POST --data-binary 'update=INSERT DATA { <https://test.com/s> <https://
test.com/p> <https://test.com/o> . }' https://your-neptune-endpoint:port/sparql
```

O exemplo anterior insere o seguinte triplo no gráfico padrão do SPARQL: `<https://test.com/s> <https://test.com/p> <https://test.com/o>`

3. Digite o seguinte para enviar um SPARQL **`QUERY`** usando HTTP `POST` e o comando curl.

```
curl -X POST --data-binary 'query=select ?s ?p ?o where {?s ?p ?o} limit 10'
 https://your-neptune-endpoint:port/sparql
```

O exemplo anterior retorna até 10 dos triplos (subject-predicate-object) no gráfico usando a consulta `?s ?p ?o` com um limite de 10. Para consultar outro elemento, substitua-a por outra consulta do SPARQL.

> **Note**
>
> O tipo MIME padrão de uma resposta é `application/sparql-results+json` para `SELECT` e consultas `ASK`.
> O tipo MIME padrão de uma resposta é `application/n-quads` para `CONSTRUCT` e consultas `DESCRIBE`.
> Para obter uma lista de todos os tipos MIME disponíveis, consulte API HTTP do SPARQL.

Carregar dados no Neptune

O Amazon Neptune fornece um processo para carregar dados de arquivos externos diretamente em uma instância de banco de dados do Neptune. Você pode usar esse processo em vez de executar um grande número de instruções `INSERT`, etapas `addV` e `addE` ou outras chamadas de API.

Veja a seguir links para informações de carregamento adicionais.

- Métodos comuns para carregar dados: Carregamento de dados
- Formatos de dados compatíveis com o carregador em massa: the section called “Formatos de dados”.
- Exemplo de carregamento: the section called “Exemplo de carregamento”

Monitorar o Amazon Neptune

O Amazon Neptune é compatível com os métodos de monitoramento a seguir.

- Amazon CloudWatch — O Amazon Neptune envia métricas automaticamente e também oferece suporte CloudWatch a alarmes. CloudWatch Para ter mais informações, consulte the section called “Usando CloudWatch”.
- AWS CloudTrail— O Amazon Neptune suporta o uso de registros de API. CloudTrail Para ter mais informações, consulte the section called “Registrando chamadas da API Neptune com AWS CloudTrail”.
- Marcação: use tags para adicionar metadados aos recursos do Neptune e monitorar o uso com base em tags. Para ter mais informações, consulte the section called “Marcar os recursos do Neptune”.
- Arquivos de log de auditoria: visualize, baixe ou observe arquivos de log de banco de dados usando o console do Neptune. Para ter mais informações, consulte the section called “Logs de auditoria com Neptune”.
- Status de instância: confira a integridade do mecanismo de banco de dados de grafos de uma instância do Neptune, descubra qual versão do mecanismo está instalada e obtenha outras informações de status do mecanismo usando a API de status da instância.

Solução de problemas e práticas recomendadas no Neptune

Os links a seguir podem ser úteis para resolver problemas com o Amazon Neptune.

- Práticas recomendadas: para ver soluções para problemas comuns e sugestões de desempenho, consulte Práticas recomendadas.
- Erros de serviço: para obter uma lista de erros para conexões de banco de dados de grafos e APIs de gerenciamento, consulte Erros do Neptune.
- Limites de serviço: para obter informações sobre os limites do Neptune, consulte Limites do Neptune.
- Versões do mecanismo: para obter informações sobre versões do mecanismo do grafo, incluindo notas de versão, consulte Versões do mecanismo.
- Fóruns de suporte: para participar de uma discussão sobre o Neptune, consulte o Fórum do Amazon Neptune.

- Preços: para obter informações sobre os custos de uso do Amazon Neptune, consulte Preços do Amazon Neptune.
- AWS Support — Para obter ajuda e orientação dos especialistas, consulte AWS Support.

Criar um banco de dados global do Neptune

Um banco de dados global do Amazon Neptune abrange várias Regiões da AWS, permitindo leituras globais de baixa latência e fornecendo recuperação rápida no caso raro de uma interrupção afetar toda uma Região da AWS.

Um banco de dados global do Neptune tem um cluster de banco de dados principal em uma região e até cinco clusters de banco de dados secundários em diferentes regiões.

As gravações só podem ocorrer na região principal. As regiões secundárias são compatíveis apenas com leituras. Cada região secundária pode ter até 16 instâncias de leitor.

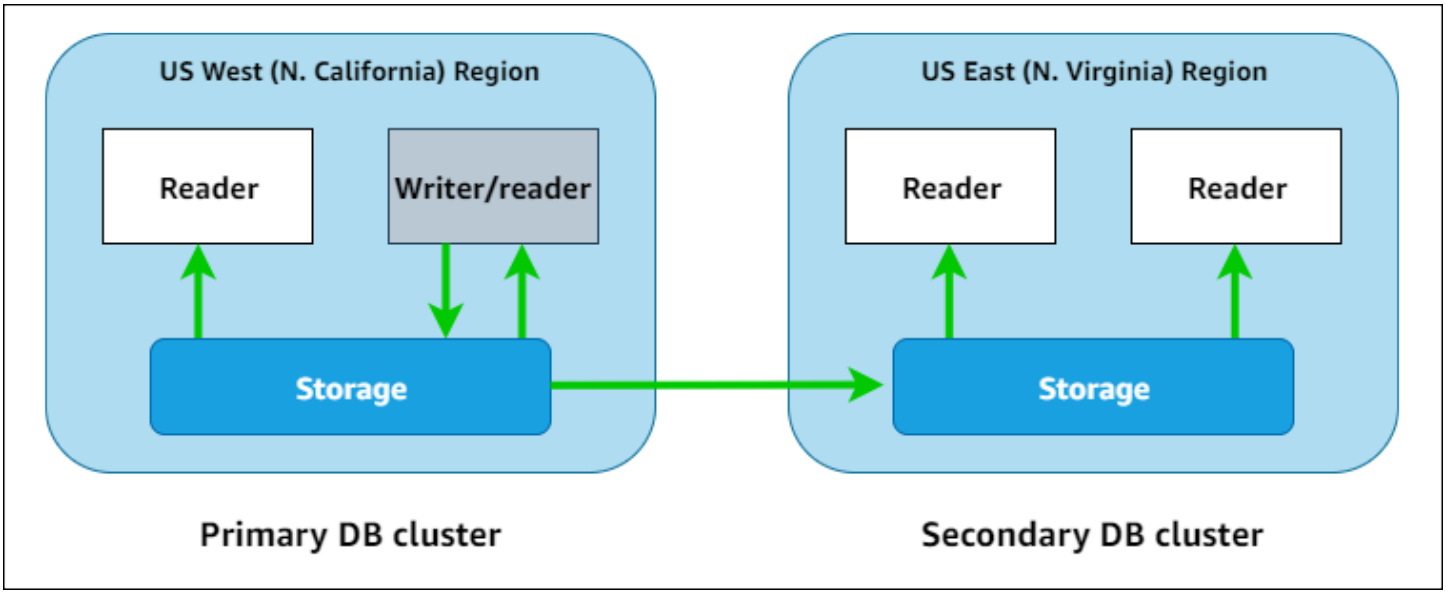
Tópicos

Visão geral de bancos de dados globais no Amazon Neptune

Ao usar um banco de dados global Neptune, é possível executar aplicações distribuídas globalmente em um único banco de dados que abranja várias Regiões da AWS.

Um banco de dados global Neptune consiste em um cluster de banco de dados em uma Região da AWS principal onde os dados são gravados e até cinco clusters de banco de dados somente leitura em Regiões da AWS secundárias. Quando você executa uma operação de gravação no cluster de banco de dados principal, o Neptune replica os dados gravados em todos os clusters de banco de dados secundários usando uma infraestrutura dedicada, normalmente com latência de menos de um segundo.

O diagrama a seguir mostra um exemplo de banco de dados global que abrange duas Regiões da AWS:



É possível escalar cada cluster secundário de maneira independente para lidar com workloads somente leitura adicionando uma ou mais instâncias de réplica de leitura.

Para realizar operações de gravação, é necessário se conectar ao endpoint do cluster de banco de dados do cluster de banco de dados principal. Somente o cluster principal pode realizar operações de gravação. Então, conforme mostrado no diagrama acima, a replicação é realizada pelo volume de armazenamento do cluster, não pelo mecanismo do banco de dados.

O banco de dados global Neptune foi criado para aplicações com uma presença mundial. Os clusters de banco de dados secundários somente leitura aceitam operações de leitura mais próximas dos usuários da aplicação.

Um banco de dados global Neptune é compatível com duas abordagens diferentes para o failover:

- Para recuperar-se de uma interrupção na região principal, use o processo manual não planejado de desanexação e promoção, em que você separa um dos clusters secundários, transformando-o em um cluster autônomo e, depois, o promove como o novo cluster principal.
- Para procedimentos operacionais planejados, como manutenção, use failover planejado gerenciado, em que você realoca o cluster principal para uma das regiões secundárias sem perda de dados.

Vantagens de usar bancos de dados globais no Amazon Neptune

O uso de banco de dados global tem as seguintes vantagens:

- Leituras globais com latência local: se você tem escritórios em todo o mundo, um banco de dados global permite que seus escritórios em regiões secundárias acessem dados na própria região com latência local.
- Clusters de banco de dados Neptune secundários escaláveis: é possível dimensionar os clusters secundários adicionando instâncias de banco de dados de réplica de leitura. Como os clusters secundários são somente leitura, cada um pode oferecer compatibilidade com até 16 réplicas de leitura em vez do limite comum de 15.
- Replicação rápida para clusters de banco de dados secundários: a replicação de clusters de banco de dados principais para secundários é rápida, com latência normalmente abaixo de um segundo, com pouco impacto no desempenho no cluster de banco de dados principal. Como a replicação é realizada no armazenamento, os recursos da instância de banco de dados estão totalmente disponíveis para workloads de leitura e gravação de aplicações.
- Recuperação de interrupções em toda a região: os clusters de banco de dados secundários permitem mover o cluster principal para uma nova região com maior rapidez, com menor RTO e menor perda de dados (menor RPO) do que as soluções tradicionais de replicação.

Limitações dos bancos de dados globais no Amazon Neptune

No momento, as seguintes limitações se aplicam aos bancos de dados globais do :

- Os bancos de dados globais Neptune estão disponíveis apenas nas seguintes Regiões da AWS:
  - Leste dos EUA (Norte da Virgínia): `us-east-1`
  - Leste dos EUA (Ohio): `us-east-2`
  - Oeste dos EUA (N. da Califórnia): `us-west-1`
  - Oeste dos EUA (Oregon): `us-west-2`
  - Europa (Irlanda): `eu-west-1`
  - Europa (Londres): `eu-west-2`
  - Ásia-Pacífico (Tóquio): `ap-northeast-1`
- No momento, os bancos de dados globais Neptune não são compatíveis com ajuste de escala automático para clusters de banco de dados secundários.
- Você não pode aplicar um grupo de parâmetros personalizado ao cluster de banco de dados global enquanto estiver executando uma atualização de versão principal desse banco de dados global. Em vez disso, crie os grupos de parâmetros personalizados em cada região do cluster global e, depois, aplique-os manualmente aos clusters regionais após a atualização.

- Não é possível interromper ou iniciar os clusters de banco de dados em um banco de dados global individualmente.
- Instâncias de réplica de leitura em um cluster de banco de dados secundário podem ser reiniciadas em determinadas circunstâncias. Se a instância de gravador do cluster principal for reiniciada ou sofrer failover, todas as instâncias em regiões secundárias também serão reiniciadas. O cluster secundário fica indisponível até que todas as instâncias estejam novamente sincronizadas com a instância de gravador do cluster de banco de dados principal.

Configurar um banco de dados global no Amazon Neptune

É possível criar um banco de dados Neptune de uma das seguintes maneiras:

- Crie um banco de dados global com todos os novos clusters de banco de dados e instâncias.
- Crie um banco de dados global usando um cluster de banco de dados Neptune existente como cluster principal.

Tópicos

Requisitos de configuração para um banco de dados global no Amazon Neptune

Um banco de dados global Neptune abrange pelo menos duas Regiões da AWS. A Região da AWS principal contém um cluster de banco de dados Neptune com uma instância de gravador. De uma a cinco Regiões da AWS secundárias contêm um cluster de banco de dados Neptune somente leitura composto inteiramente de instâncias de réplica de leitura. Pelo menos uma Região da AWS secundária é necessária.

Os clusters de banco de dados Neptune que compõem um banco de dados global têm os seguintes requisitos específicos:

- Requisitos de classe de instância de banco de dados: um banco de dados global requer as classes de instância `r5` ou `r6g` de banco de dados otimizadas para workloads de uso intenso de memória, como um tipo de instância `db.r5.large`.
- Requisitos da Região da AWS: um banco de dados global precisa de um cluster de banco de dados Neptune principal em uma Região da AWS e pelo menos um cluster de banco de dados Neptune secundário em uma região diferente. É possível criar até cinco clusters de banco de dados secundários do Neptune somente leitura e cada um deve estar em uma região diferente. Em outras palavras, dois clusters de banco de dados Neptune em um banco de dados global Neptune não podem estar na mesma Região da AWS.
- Requisitos de versão do mecanismo: a versão do mecanismo do Neptune usada por todos os clusters de banco de dados no banco de dados global deve ser a mesma e deve ser superior ou igual a `1.2.0.0`. Se você não especificar a versão do mecanismo ao criar um banco de dados global, um cluster ou uma instância, a versão mais recente do mecanismo será usada.

> **Important**
>
> Embora os grupos de parâmetros do cluster de banco de dados possam ser configurados de forma independente para cada cluster de banco de dados em um banco de dados global, é melhor manter as configurações consistentes em todos os clusters para evitar alterações inesperadas de comportamento se um cluster secundário precisar ser promovido a principal. Por exemplo, use as mesmas configurações para índices de objetos, fluxos, etc. em todos os clusters de banco de dados.

Usar a AWS CLI para criar um banco de dados global no Amazon Neptune

> **Note**
>
> Os exemplos nesta seção seguem a convenção UNIX de usar uma barra invertida (\) como caractere extensor de linha. No Windows, substitua a barra invertida por um acento circunflexo (^).

Como criar um banco de dados global usando a AWS CLI

1. Comece criando um banco de dados global vazio usando o comando `create-global-cluster` da AWS CLI (que envolve a API CreateGlobalCluster). Especifique o nome da Região da AWS que você deseja que seja principal, defina o Neptune como o mecanismo do banco de dados e, opcionalmente, especifique a versão do mecanismo que você deseja usar (deve ser a versão 1.2.0.0 ou posterior):

```
aws neptune create-global-cluster
  --region (primary region, such as us-east-1) \
  --global-cluster-identifier (ID for the global database) \
  --engine neptune \
  --engine-version (engine version; this is optional)
```

2. Pode levar alguns minutos para que o banco de dados global esteja disponível. Portanto, antes de ir para a próxima etapa, use o comando `describe-global-clusters` da CLI (que envolve a API DescribeGlobalClusters) para conferir se o banco de dados global está disponível:

```
aws neptune describe-global-clusters \
  --region (primary region) \
  --global-cluster-identifier (global database ID)
```

3. Depois que o banco de dados global Neptune estiver disponível, você poderá criar um cluster de banco de dados Neptune para ser o cluster principal:

```
aws neptune create-db-cluster \
  --region (primary region) \
  --db-cluster-identifier (ID for the primary DB cluster) \
  --engine neptune \
  --engine-version (engine version; must be >= 1.2.0.0) \
  --global-cluster-identifier (global database ID)
```

4. Use o comando `describe-db-clusters` da AWS CLI para confirmar se o novo cluster de banco de dados está pronto para você adicionar a instância de banco de dados principal:

```
aws neptune describe-db-clusters \
  --region (primary region) \
  --db-cluster-identifier (primary DB cluster ID)
```

   Quando os resultados mostrarem `"Status": "available"`, siga para a próxima etapa.

5. Crie a instância de banco de dados principal para o cluster principal usando o comando `create-db-instance` da AWS CLI. É necessário usar um dos tipos de instância `r5` ou `r6g` otimizados para memória, como `db.r5.large`.

```
aws neptune create-db-instance \
  --region (primary region) \
  --db-cluster-identifier (primary cluster ID) \
  --db-instance-class (instance class) \
  --db-instance-identifier (ID for the DB instance) \
  --engine neptune \
  --engine-version (optional: engine version)
```

> **Note**
>
> Se você planeja adicionar dados ao novo cluster de banco de dados principal usando o carregador em massa do Neptune, faça isso antes de adicionar regiões secundárias. Isso é mais rápido e econômico do que realizar um carregamento em massa depois que o banco de dados global estiver totalmente configurado.

Agora, adicione uma ou mais regiões secundárias ao novo banco de dados global, conforme descrito em Adicionar uma região secundária usando a AWS CLI.

Transformar um cluster de banco de dados existente em um banco de dados global

Para transformar um cluster de banco de dados existente em um banco de dados global, use o comando `create-global-cluster` da AWS CLI para criar um banco de dados global na mesma Região da AWS onde o cluster de banco de dados existente está localizado e defina o parâmetro `--source-db-cluster-identifier` como o nome do recurso da Amazon (ARN) do cluster existente no local:

```
aws neptune create-global-cluster \
  --region (region where the existing cluster is located) \
  --global-cluster-identifier (provide an ID for the new global database) \
  --source-db-cluster-identifier (the ARN of the existing DB cluster) \
  --engine neptune \
  --engine-version (engine version; this is optional)
```

Agora, adicione uma ou mais regiões secundárias ao novo banco de dados global, conforme descrito em Adicionar uma região secundária usando a AWS CLI.

Usar um cluster de banco de dados restaurado a partir de um snapshot como cluster principal

É possível transformar um cluster de banco de dados restaurado de um snapshot em um banco de dados global Neptune. Depois que a restauração for concluída, transforme o cluster de banco de dados criado no cluster principal de um novo banco de dados global conforme descrito acima.

Adicionar regiões secundárias de um banco de dados global a uma região principal no Amazon Neptune

Um banco de dados global Neptune precisa de pelo menos um cluster de banco de dados secundário do Neptune em uma Região da AWSdiferente do cluster de banco de dados principal. Você pode anexar até cinco clusters de banco de dados secundários ao cluster de banco de dados principal.

Cada cluster de banco de dados secundário adicionado reduz em um o número máximo de instâncias de réplica de leitura que você pode ter no cluster principal. Por exemplo, se houver quatro clusters secundários, o número máximo de instâncias de réplica de leitura que você poderá ter no cluster principal é `15 - 4 = 11`. Isso significa que, se você tiver 14 instâncias de leitor no cluster de banco de dados principal e um cluster secundário, não será possível adicionar outro cluster secundário.

Usar a AWS CLI para adicionar uma região secundária a um banco de dados global no Neptune

Como adicionar uma Região da AWS secundária a um banco de dados global Neptune usando a AWS CLI

1. Use o comando `create-db-cluster` da AWS CLI para criar um cluster de banco de dados em uma região diferente da do cluster principal e defina o parâmetro `--global-cluster-identifier` para especificar o ID do banco de dados global:

```
aws neptune create-db-cluster \
  --region (the secondary region) \
  --db-cluster-identifier (ID for the new secondary DB cluster) \
```

```
  --global-cluster-identifier (global database ID)
  --engine neptune \
  --engine-version (optional: engine version)
```

2. Use o comando `describe-db-clusters` da AWS CLI para confirmar se o novo cluster de banco de dados está pronto para você adicionar a instância de banco de dados principal:

```
aws neptune describe-db-clusters \
  --region (primary region) \
  --db-cluster-identifier (primary DB cluster ID)
```

   Quando os resultados mostrarem `"Status": "available"`, siga para a próxima etapa.

3. Crie a instância de banco de dados principal para o cluster principal por meio do comando `create-db-instance` da AWS CLI, usando um tipo de instância na classe de instância `r5` ou `r6g`:

```
aws neptune create-db-instance \
  --region (secondary region) \
  --db-cluster-identifier (secondary cluster ID) \
  --db-instance-class (instance class) \
  --db-instance-identifier (ID for the DB instance) \
  --engine neptune \
  --engine-version (optional: engine version)
```

> **Note**
>
> Se você não pretende atender a um grande número de solicitações de leitura na região secundária e se preocupa principalmente em manter o backup confiável dos dados, é possível criar um cluster de banco de dados secundário sem instâncias de banco de dados. Isso economiza dinheiro, já que você pagará apenas pelo armazenamento do cluster secundário, que o Neptune manterá sincronizado com o armazenamento no cluster de banco de dados principal.

Conectar-se a um banco de dados global Neptune

A forma de conexão com um banco de dados global Neptune depende se você precisa gravar ou ler nele:

- Para solicitações ou consultas somente leitura, conecte-se ao endpoint de leitor do cluster do Neptune na Região da AWS.
- Para executar consultas de mutação, conecte-se ao endpoint do cluster de banco de dados principal, que pode estar em uma Região da AWS diferente da aplicação.

Gerenciar um banco de dados global do Amazon Neptune

Com exceção do processo de failover planejado gerenciado, você pode executar a maioria das operações de gerenciamento nos clusters individuais que compõem um banco de dados global do Neptune. O processo de failover planejado gerenciado está disponível apenas para bancos de dados globais do Neptune, não para clusters de banco de dados do Neptune individuais. Para saber mais, consulte Realizar failovers planejados gerenciados para bancos de dados globais do Neptune.

Para recuperar um banco de dados global do Neptune de uma interrupção não planejada na região principal, consulte Desanexe e promova um banco de dados global do Neptune no caso de uma interrupção não planejada.

Embora os grupos de parâmetros do cluster de banco de dados possam ser configurados de forma independente para cada cluster do Neptune em um banco de dados global, é melhor manter as configurações consistentes em todos os clusters para evitar alterações inesperadas de comportamento se um cluster secundário precisar ser promovido a principal. Por exemplo, use as mesmas configurações para índices de objetos, fluxos, etc. em todos os clusters de banco de dados.

Remover um cluster de banco de dados de um banco de dados global do Neptune

Há vários motivos pelos quais convém remover um cluster de banco de dados de um banco de dados global. Por exemplo:

- Se o cluster principal ficar degradado ou isolado, você poderá removê-lo do banco de dados global e ele se tornará um cluster provisionado autônomo que pode ser usado para criar um banco de dados global (consulte Desanexe e promova um banco de dados global do Neptune no caso de uma interrupção não planejada).
- Se quiser excluir um banco de dados global, primeiro remova (desanexe) todos os clusters associados, deixando o principal por último (consulte Excluir um banco de dados global do Neptune).

É possível usar o comando da CLI remove-from-global-cluster (que envolve a API RemoveFromGlobalCluster) para separar um cluster de banco de dados do Neptune de um banco de dados global:

```
aws neptune remove-from-global-cluster \
  --region (region of the cluster to remove) \
  --global-cluster-identifier (global database ID) \
  --db-cluster-identifier (ARN of the cluster to remove)
```

O cluster de banco de dados desanexado se torna um cluster de banco de dados autônomo.

Excluir um banco de dados global do Neptune

Não é possível excluir um banco de dados global e os clusters associados em uma única etapa. Em vez disso, você precisa excluir os componentes um por um:

1. Desanexe todos os clusters de banco de dados secundários do banco de dados global, conforme descrito em Remover um cluster. Se quiser, você agora poderá excluí-los individualmente.
2. Desanexe o cluster do banco de dados principal do banco de dados global.
3. Exclua todas as instâncias de banco de dados de réplica de leitura do cluster principal.
4. Exclua a instância de banco de dados principal (de gravador) do cluster principal. Se você fizer isso no console, ele também excluirá o cluster de banco de dados.
5. Crie o próprio banco de dados global. Para fazer isso com a AWS CLI, use o comando da CLI delete-global-cluster (que encapsula a API DeleteGlobalCluster) da seguinte forma:

   ```
   aws neptune delete-global-cluster \
     --region (region of the DB cluster to delete) \
     --global-cluster-identifier (global database ID)
   ```

Modificar um banco de dados global do Neptune

Os grupos de parâmetros do cluster de banco de dados podem ser configurados de forma independente para cada cluster de banco de dados do Neptune em um banco de dados global, mas é melhor manter as configurações consistentes em todos os clusters para evitar alterações inesperadas de comportamento se um cluster secundário precisar ser promovido a principal.

É possível modificar as configurações do próprio banco de dados global usando o comando da CLI modify-global-cluster (que encapsula a API ModifyGlobalCluster). Por exemplo, é possível alterar o

identificador de banco de dados global e, ao mesmo tempo, desativar a proteção contra exclusão desta forma:

```
aws neptune modify-global-cluster \
  --region (region of the DB cluster to modify) \
  --global-cluster-identifier (current global database ID) \
  --new-global-cluster-identifier (new global database ID to assign) \
  --deletion-protection false
```

Usar failover em um banco de dados global do Neptune

Um banco de dados global do Neptune oferece recursos de failover mais abrangentes do que um cluster de banco de dados do Neptune autônomo. Usando um banco de dados global, é possível planejar e se recuperar de desastres com rapidez. A recuperação de desastres é geralmente avaliada usando a avaliação do objetivo de tempo de recuperação (RTO) e do objetivo de ponto de recuperação (RPO):

- Objetivo de tempo de recuperação (RTO): é a rapidez com que um sistema retorna ao estado funcional após um desastre. Em outras palavras, o RTO mede o tempo de inatividade. Para um banco de dados global do Neptune, o RTO pode estar na ordem dos minutos.
- Objetivo de ponto de recuperação (RPO): a quantidade de tempo durante o qual os dados estão sendo perdidos. Para um banco de dados global do Neptune, o RPO é normalmente medido em segundos (consulte Realizar failovers planejados gerenciados para bancos de dados globais do Neptune).

Em um banco de dados global do Neptune, há duas abordagens diferentes para o failover:

- Desanexar e promover (recuperação não planejada): para recuperar de uma interrupção não planejada ou realizar testes de recuperação de desastres (testes de DR), realize uma desanexação e promoção entre regiões em um dos clusters de banco de dados secundários no banco de dados global. O RTO deste processo manual depende da rapidez com que você pode executar as tarefas listadas em Desanexar e promover. Normalmente, o RPO corresponde a vários segundos, mas isso depende do atraso de replicação do armazenamento em toda a rede no momento da falha.
- Failover planejado gerenciado: essa abordagem se destina à manutenção operacional e a outros procedimentos operacionais planejados, como realocar o cluster de banco de dados principal do banco de dados global para uma das regiões secundárias. Como esse processo sincroniza

clusters de banco de dados secundários com o principal antes de fazer outras alterações, o RPO é efetivamente 0 (ou seja, sem perda de dados). Consulte Realizar failovers planejados gerenciados para bancos de dados globais do Neptune.

Desanexe e promova um banco de dados global do Neptune no caso de uma interrupção não planejada

Na situação muito rara em que o banco de dados global do Neptune sofra uma interrupção inesperada na Região da AWS principal, o cluster de banco de dados principal do Neptune e o nó de gravador ficam indisponíveis, e a replicação entre o cluster principal e os secundários cessará. Para minimizar o tempo de inatividade (RTO) e a perda de dados (RPO), realize rapidamente uma desanexação e promoção entre regiões para reconstruir o banco de dados global.

> **Tip**
>
> É uma boa ideia entender esse processo antes de usá-lo e ter um plano para agir rapidamente ao primeiro sinal de um problema em toda a região.
>
> - Use o Amazon CloudWatch regularmente para monitorar os tempos de atraso dos clusters secundários para que você possa identificar a região secundária com o menor tempo de atraso se precisar fazer failover.
> - Teste o plano para conferir se os procedimentos estão completos e precisos.
> - Use um ambiente simulado para garantir que a equipe esteja treinada e pronta para realizar um failover de DR rapidamente, caso seja necessário.

Como fazer failover para um cluster secundário após uma interrupção não planejada na região principal

1. Pare de emitir consultas de mutação e outras operações de gravação no cluster de banco de dados principal.
2. Identifique um cluster de banco de dados em uma Região da AWS secundária para usar como o novo cluster de banco de dados principal do banco de dados global. Se o banco de dados global tiver duas ou mais Regiões da AWS secundárias, selecione o cluster secundário com o menor tempo de atraso.

3. Desanexe o cluster de banco de dados secundário que você escolheu do banco de dados global do Neptune.

   A remoção de um cluster de banco de dados secundário de um banco de dados global do Neptune interrompe imediatamente a replicação de dados do principal para esse secundário e promove-o ao cluster de banco de dados autônomo com recursos completos de leitura/gravação. Todos os outros clusters secundários no banco de dados global ainda estarão disponíveis e poderão aceitar chamadas de leitura da aplicação.

   Antes de recriar o banco de dados global do Neptune, você também precisará desanexar os outros clusters secundários para evitar inconsistências de dados entre os clusters (consulte Remover um cluster).

4. Reconfigure a aplicação para enviar todas as operações de gravação ao cluster de banco de dados autônomo do Neptune que você escolheu para se tornar o novo cluster principal, usando o novo endpoint. Se você aceitou os nomes padrão ao criar o banco de dados global do Neptune, poderá alterar o endpoint removendo `-ro` da string do endpoint do cluster na aplicação.

   Por exemplo, o endpoint do cluster secundário `my-global.cluster-**ro**-aaaaaabbbbbb.us-west-1.neptune.amazonaws.com` se torna `my-global.cluster-aaaaaabbbbbb.us-west-1.neptune.amazonaws.com` quando esse cluster é separado do banco de dados global.

   Esse cluster de banco de dados do Neptune se torna o cluster principal de um novo banco de dados global do Neptune quando você começa a adicionar regiões a ele na próxima etapa.

5. Adicione uma Região da AWS ao cluster de banco de dados. Quando você faz isso, o processo de replicação de primário para secundário começa. Consulte Adicionar regiões secundárias de um banco de dados global a uma região principal no Amazon Neptune .

6. Adicione mais Regiões da AWS conforme necessário para recriar a topologia necessária para oferecer suporte à aplicação.

Certifique-se de que as gravações de aplicações sejam enviadas ao cluster de banco de dados correto do Neptune antes, durante e depois de fazer essas alterações. Isso evita inconsistências de dados entre os clusters de banco de dados no banco de dados global do Neptune (são conhecidos como problemas de cérebro dividido).

Realizar failovers planejados gerenciados para bancos de dados globais do Neptune

O failover planejado gerenciado permite realocar o cluster principal do banco de dados global do Neptune para uma Região da AWS diferente sempre que desejar. Algumas organizações vão querer trocar os locais do cluster principal regularmente.

> **Note**
>
> O processo de failover planejado gerenciado descrito aqui foi projetado para ser usado em um banco de dados global íntegro do Neptune. Para se recuperar de uma interrupção não planejada ou realizar testes de recuperação de desastres (DR), siga o processo de desanexar e promover.

Durante um failover planejado gerenciado, é realizado failover do cluster principal para a região secundária escolhida enquanto a topologia de replicação existente do banco de dados global é preservada. Antes do início do processo de failover planejado gerenciado, o banco de dados global sincroniza todos os clusters secundários com o cluster principal. Depois de garantir que todos os clusters sejam sincronizados, o failover planejado gerenciado começa. O cluster de banco de dados na região principal se torna somente leitura, e o cluster secundário escolhido promove uma de suas instâncias somente leitura para o status completo de gravador, permitindo assim que o cluster assuma a função de cluster principal. Como todos os clusters secundários foram sincronizados com o principal no início do processo, o novo principal continua as operações para o banco de dados global sem perder dados. O banco de dados fica indisponível por um curto período enquanto os clusters principais e secundários selecionados estão assumindo as novas funções.

Para otimizar a disponibilidade da aplicação, realize o failover em períodos de pouca demanda, quando as gravações no cluster de banco de dados principal forem mínimas. Além disso, realize as seguintes etapas antes de iniciar o failover:

- Coloque as aplicações off-line sempre que possível para reduzir as gravações no cluster principal.
- Confira os tempos de atraso de todos os clusters de banco de dados do Neptune secundários no banco de dados global e selecione o secundário com o menor tempo de atraso geral para se tornar o principal. Use Amazon CloudWatch para visualizar a `NeptuneGlobalDBProgressLag` métrica de todos os secundários. Essa métrica informa o quão longe um secundário está atrás do cluster de banco de dados principal, em milissegundos. O valor é diretamente proporcional ao

tempo que o Neptune levará para concluir o failover. Em outras palavras, quanto maior o valor de atraso, maior será a interrupção provocada pelo failover, então escolha a região com o menor atraso. Consulte Métricas de Neptune CloudWatch para obter mais informações.

Durante um failover planejado gerenciado, o cluster de banco de dados secundário escolhido é promovido à nova função como principal, mas não herda a configuração completa do cluster de banco de dados principal. Uma incompatibilidade na configuração pode levar a problemas de performance, incompatibilidades de workload e outros comportamentos anômalos. Para evitar esses problemas, resolva os seguintes tipos de diferença de configuração entre clusters de banco de dados global antes do failover:

- Configure os parâmetros no novo principal para que correspondam ao principal atual.
- Configure ferramentas, opções e alarmes de monitoramento: configure o cluster de banco de dados que será o novo principal com a mesma capacidade de registro em log, alarmes, etc. que o principal atual tem.
- Configure integrações a outros serviços da AWS: se o banco de dados global do Neptune se integrar a serviços da AWS, como o AWS Identity and Access Management (IAM), o Amazon S3 ou o AWS Lambda, não se esqueça de configurá-los conforme necessário para integrá-los ao novo cluster de banco de dados principal.

Quando o processo de failover for concluído e o cluster de banco de dados promovido estiver pronto para lidar com as operações de gravação no banco de dados global, altere as aplicações para usar o novo endpoint para o novo principal.

Usar a AWS CLI para iniciar o failover planejado gerenciado

Use o comando da CLI failover-global-cluster (que envolve a API FailoverGlobalCluster) para realizar failover do banco de dados global do Neptune:

```
aws neptune failover-global-cluster \
  --region (the region where the primary cluster is located) \
  --global-cluster-identifier (global database ID) \
  --target-db-cluster-identifier (the ARN of the secondary DB cluster to promote)
```

> **Note**
>
> A API `failover-global-cluster` não está disponível na visualização. Ela fará parte do lançamento do GA.

Monitorar um banco de dados global do Neptune usando métricas do CloudWatch

O Neptune é compatível com as seguintes métricas do CloudWatch que você pode usar para monitorar um banco de dados global do Neptune:

- **`GlobalDbDataTransferBytes`**: o número de bytes de dados de redo log transferidos da Região da AWS principal para a Região da AWS secundária em um banco de dados global do Neptune.
- **`GlobalDbReplicatedWriteIO`**: o número de operações de E/S de gravação replicadas da Região da AWS principal no banco de dados global para o volume do cluster em uma Região da AWS secundária.

  Os cálculos de cobrança para cada cluster de banco de dados em um banco de dados global do Neptune usam a métrica `VolumeWriteIOPS` para contabilizar as gravações realizadas nesse cluster. Para o cluster de banco de dados principal, os cálculos de cobrança usam `NeptuneGlobalDbReplicatedWriteIO` para contabilizar a replicação entre regiões para clusters de banco de dados secundários.
- **`GlobalDbProgressLag`**: o número de milissegundos em que um cluster secundário está atrás do cluster principal para transações de usuário e transações do sistema.

Métrica	Dimensão	Publicado em	Unidades
`GlobalDbDataTransferBytes`	SourceRegion, DBClusterIdentifier	Secondary	Bytes
`GlobalDbReplicatedWriteIO`	SourceRegion, DBClusterIdentifier	Secondary	Contagem

Métrica	Dimensão	Publicado em	Unidades
`GlobalDbProgressLag`	DBClusterIdentifier, SecondaryRegion: no DBClusterIdentifier principal, SourceRegion: no secundário	Primário, secundário	Milissegundos

Visão geral dos atributos do Amazon Neptune

Esta seção fornece uma visão geral dos atributos específicos do Neptune, incluindo:

- Conformidade do Neptune com os padrões de linguagem de consulta.
- Modelo de dados do grafo do Netuno.
- Uma explicação da semântica de transações do Neptune.
- Uma introdução aos clusters e às instâncias do Neptune.
- Armazenamento, confiabilidade e disponibilidade do Neptune.
- Uma explicação dos endpoints do Netuno.
- Como os IDs de consulta personalizados do Neptune permitem que você confira o status da consulta.
- Usar o modo de laboratório do Neptune para habilitar atributos experimentais.
- Uma descrição do mecanismo DFE do Neptune.
- Conectividade JDBC do Neptune.
- Uma lista dos lançamentos do mecanismo do Neptune e como atualizar o mecanismo.

> **Note**
>
> Esta seção não aborda o uso das linguagens de consulta que você pode usar para acessar os dados em um grafo do Neptune.
> Para obter informações sobre como se conectar a um cluster de banco de dados do Neptune com o Gremlin, consulte Acessar o grafo do Neptune com o Gremlin.
> Para obter informações sobre como se conectar a um cluster de banco de dados do Neptune com o openCypher, consulte Acessar o grafo do Neptune com o openCypher.
> Para obter informações sobre como se conectar a um cluster de banco de dados do Neptune em execução, consulte Acessar o grafo do Neptune com o SPARQL.

Tópicos

- Observações sobre conformidade com os padrões do Amazon Neptune
- Modelo de dados de grafo do Neptune
- O cache de pesquisa do Neptune pode acelerar as consultas de leitura

- Semântica de transação no Neptune
- Clusters e instâncias de banco de dados do Amazon Neptune
- Armazenamento, confiabilidade e disponibilidade do Amazon Neptune.
- Conectar-se a endpoints do Amazon Neptune
- Injetar um ID personalizado em uma consulta do Gremlin ou do SPARQL no Neptune
- Modo de laboratório do Neptune
- O mecanismo de consulta alternativo (DFE) do Amazon Neptune
- Gerenciar estatísticas a serem utilizadas pelo DFE do Neptune
- Obter um relatório resumido rápido sobre o grafo
- Conectividade JDBC do Amazon Neptune
- Atualizações do mecanismo do Amazon Neptune

Observações sobre conformidade com os padrões do Amazon Neptune

O Amazon Neptune está em conformidade com os padrões aplicáveis na implementação das linguagens de consulta do grafo Gremlin e SPARQL na maioria dos casos.

Estas seções descrevem os padrões, bem como as áreas em que o Neptune os amplia ou diverge deles.

Tópicos

Conformidade com os padrões do Gremlin no Amazon Neptune

As seções a seguir fornecem uma visão geral da implementação do Gremlin no Neptune e como ela difere da implementação do Apache. TinkerPop

O Neptune implementa algumas etapas do Gremlin nativamente em seu mecanismo e usa a implementação do TinkerPop Apache Gremlin para processar outras (consulte). Suporte nativo para etapas do Gremlin no Amazon Neptune

> **Note**
>
> Para obter exemplos concretos dessas diferenças de implementação mostradas no console do Gremlin e no Amazon Neptune, consulte a seção the section called “Usar o Gremlin” do Quick Start.

Tópicos

- Execução do script
- Sessões
- Transações
- IDs de vértice e de borda
- IDs fornecidos pelo usuário
- IDs de propriedades de vértice
- Cardinalidade de propriedades de vértice
- Atualizar uma propriedade de vértice
- Rótulos
- Caracteres de escape
- Limitações do Groovy
- Serialização
- Etapas do Lambda
- Métodos do Gremlin não compatíveis
- Etapas do Gremlin não compatíveis
- Atributos do grafo do Gremlin no Neptune

Padrões aplicáveis para Gremlin

- A linguagem Gremlin é definida pela TinkerPop documentação do Apache e pela TinkerPop implementação do Gremlin pelo Apache, e não por uma especificação formal.
- Para formatos numéricos, o Gremlin segue o padrão IEEE 754 (IEEE 754-2019 - IEEE Standard for Floating-Point Arithmetic. Para obter mais informações, consulte também a página do IEEE 754 na Wikipédia).

Variáveis e parâmetros em scripts

No que diz respeito às variáveis pré-vinculadas, o objeto de percurso `g` é pré-vinculado no Neptune, e o objeto `graph` não é compatível.

Embora o Neptune não seja compatível com variáveis do Gremlin nem com a parametrização em scripts, é possível encontrar exemplos de scripts para o Gremlin Server na Internet que contêm declarações de variáveis, como:

```
String query = "x = 1; g.V(x)";
List<Result> results = client.submit(query).all().get();
```

Também há muitos exemplos que usam parametrização (ou vinculações) ao enviar consultas, como:

```
Map<String,Object> params = new HashMap<>();
params.put("x",1);
String query = "g.V(x)";
List<Result> results = client.submit(query).all().get();
```

Os exemplos de parâmetro geralmente são associados a avisos sobre penalidades de desempenho por não parametrizar quando possível. Você pode encontrar muitos exemplos desse tipo, e todos parecem bastante convincentes sobre a necessidade de parametrizar. TinkerPop

No entanto, tanto o recurso de declaração de variáveis quanto o recurso de parametrização (junto com os avisos) só se aplicam ao Gremlin Server quando TinkerPop ele está usando o. `GremlinGroovyScriptEngine` Eles não se aplicam quando o Gremlin Server usa a gramática `gremlin-language` ANTLR do Gremlin para analisar consultas. A gramática ANTLR não aceita declarações de variáveis nem parametrização, portanto, ao usar o ANTLR, você não precisa se preocupar em deixar de parametrizar. Como a gramática ANTLR é um componente mais recente TinkerPop, o conteúdo mais antigo que você pode encontrar na Internet geralmente não reflete essa distinção.

O Neptune usa a gramática ANTLR no mecanismo de processamento de consultas em vez do `GremlinGroovyScriptEngine`, portanto, não é compatível com variáveis, parametrização nem com a propriedade `bindings`. Como resultado, os problemas relacionados à falha na parametrização não se aplicam ao Neptune. Usando o Neptune, é perfeitamente seguro simplesmente enviar a consulta como está, onde normalmente seria parametrizada. Como resultado, o exemplo anterior pode ser simplificado sem penalidades de desempenho da seguinte forma:

```
String query = "g.V(1)";
List<Result> results = client.submit(query).all().get();
```

TinkerPop enumerações

O Neptune não é compatível com nomes de classes totalmente qualificados para valores de enumeração. Por exemplo, você deve usar `single` e não `org.apache.tinkerpop.gremlin.structure.VertexProperty.Cardinality.single` na solicitação do Groovy.

O tipo de enumeração é determinado pelo tipo do parâmetro.

A tabela a seguir mostra os valores de enumeração permitidos e o nome TinkerPop totalmente qualificado relacionado.

Valores permitidos	Classe
`id`, `key`, `label`, `value`	org.apache.tinkerpop.gremlin.structure.T
`T.id`, `T.key`, `T.label`, `T.value`	org.apache.tinkerpop.gremlin.structure.T
`set`, `single`	org.apache.tinkerpop.gremlin.structure.VertexProperty.Cardinalidade
`asc`, `desc`, `shuffle`	org.apache.tinkerpop.gremlin.process.traversal.Order
`Order.asc` , `Order.desc` , `Order.shuffle`	org.apache.tinkerpop.gremlin.process.traversal.Order
`global`, `local`	org.apache.tinkerpop.gremlin.process.traversal.Scope
`Scope.global` , `Scope.local`	org.apache.tinkerpop.gremlin.process.traversal.Scope
`all`, `first`, `last`, `mixed`	org.apache.tinkerpop.gremlin.process.traversal.Pop
`normSack`	org.apache.tinkerpop.gremlin.process.traversal.SackFunctions.Barreira
`addAll`, `and`, `assign`, `div`, `max`, `min`, `minus`, `mult`, `or`, `sum`, `sumLong`	org.apache.tinkerpop.gremlin.process.traversal.Operator
`keys`, `values`	org.apache.tinkerpop.gremlin.structure.Column
`BOTH`, `IN`, `OUT`	org.apache.tinkerpop.gremlin.structure.Direction

any, none	org.apache.tinkerpop.gremlin.process.traversal.step. TraversalOptionParent.Pick

Código Java

O Neptune não é compatível com chamadas para métodos definidos por chamadas arbitrárias do Java ou da biblioteca Java que não sejam as APIs do Gremlin compatíveis. Por exemplo `java.lang.*`, `Date()` e `g.V().tryNext().orElseGet()` não são permitidos.

Propriedades dos elementos

Neptune não suporta `materializeProperties` o sinalizador que foi introduzido TinkerPop na versão 3.7.0 para retornar propriedades em elementos. Como resultado, Netuno ainda retornará apenas vértices ou arestas como referências apenas com sua mão. `id` `label`

Execução do script

Todas as consultas devem começar com g, o objeto de percurso.

Em envios de consulta de string, é possível emitir vários percursos separados por ponto-e-vírgula (`;`) ou um caractere de nova linha (\n). Para ser executada, cada instrução que não seja a última deve terminar com uma etapa `.iterate()`. Somente os dados de percurso final são retornados. Observe que isso não se aplica aos envios de ByteCode consultas GLV.

Sessões

As sessões no Neptune se limitam a apenas dez minutos de duração. Consulte Sessões baseadas em script do Gremlin a Referência da TinkerPop sessão para obter mais informações.

Transações

O Neptune abre uma nova transação no início de cada percurso do Gremlin e fecha a transação após a conclusão bem-sucedida do percurso. A operação é revertida quando há um erro.

Várias instruções separadas por um ponto-e-vírgula (`;`) ou um caractere de nova linha (\n) são incluídos em uma única transação. Cada instrução diferente da última deve terminar com uma etapa `next()` a ser executada. Somente os dados de percurso final são retornados.

A lógica da transação manual que usa `tx.commit()` e `tx.rollback()` não é compatível.

> **Important**
>
> Isso se aplica somente a métodos nos quais você envia a consulta do Gremlin como uma string de texto (consulte Transações do Gremlin).

IDs de vértice e de borda

Os IDs de vértice e borda do Gremlin no Neptune devem ser do tipo `String`. Essas strings de ID são compatíveis com caracteres Unicode e não podem exceder 55 MB de tamanho.

Os IDs fornecidos pelo usuário são compatíveis, mas são opcionais em uso normal. Se você não fornecer um ID ao adicionar um vértice ou uma borda, o Neptune vai gerar um UUID e convertê-lo em uma string, da seguinte forma: `"48af8178-50ce-971a-fc41-8c9a954cea62"`. Esses UUIDs não estão em conformidade com o padrão RFC, portanto, se você precisar de UUIDs padrão, deverá gerá-los externamente e fornecê-los ao adicionar vértices ou bordas.

> **Note**
>
> O comando `Load` do Neptune exige que você forneça IDs usando o campo ~id; no formato CSV do Neptune.

IDs fornecidos pelo usuário

Os IDs fornecidos pelo usuário são permitidos no Gremlin do Neptune com as condições a seguir.

- Os IDs fornecidos são opcionais.
- Somente vértices e pontos são compatíveis.
- Somente o tipo `String` é compatível.

Para criar um novo vértice com um ID personalizado, use a etapa `property` com a palavra-chave `id`: `g.addV().property(id, 'customid')`.

> **Note**
>
> Não coloque aspas em torno da palavra-chave `id`. Ela se refere a `T.id`.

Todos os IDs de vértice devem ser exclusivos, e todos os IDs de presença devem ser exclusivos. O Neptune, no entanto, permite que um vértice e uma borda tenham o mesmo ID.

Se você tentar criar um novo vértice usando o `g.addV()` e já existir um vértice com esse ID, haverá falha na operação. A exceção para isso é que, se você especificar um novo rótulo para o vértice, a operação terá êxito, mas adiciona o novo rótulo e quaisquer propriedades adicionais especificadas ao vértice existente. Nada é substituído. Um novo vértice não é criado. O ID do vértice não altera e permanece exclusivo.

Por exemplo, os comandos a seguir do Gremlin Console serão bem-sucedidos:

```
gremlin> g.addV('label1').property(id, 'customid')
gremlin> g.addV('label2').property(id, 'customid')
gremlin> g.V('customid').label()
==>label1::label2
```

IDs de propriedades de vértice

Os IDs de propriedades de vértice são gerados automaticamente e podem ser exibidos como números positivos ou negativos quando consultados.

Cardinalidade de propriedades de vértice

O Neptune é compatível com a cardinalidade set e a cardinalidade single. Se não estiver especificado, a cardinalidade set será selecionada. Isso significa que, se você definir um valor para a propriedade, um novo valor será adicionado à propriedade, mas somente se ela ainda estiver exibida no conjunto de valores. Esse é o valor da enumeração do Gremlin de Set.

Não há suporte ao `List`. Para obter mais informações sobre a cardinalidade da propriedade, consulte o tópico Vertex no Gremlin. JavaDoc

Atualizar uma propriedade de vértice

Para atualizar o valor de uma propriedade sem adicionar mais um valor ao conjunto de valores, especifique cardinalidade `single` na etapa `property`.

```
g.V('exampleid01').property(single, 'age', 25)
```

Isso remove todos os valores existentes da propriedade.

Rótulos

O Neptune é compatível com vários rótulos para um vértice. Quando cria um rótulo, você pode especificar vários rótulos separados com `::`. Por exemplo, `g.addV("Label1::Label2::Label3")` adiciona um vértice com três diferentes rótulos. A etapa `hasLabel` corresponde esse vértice com qualquer um destes três rótulos: `hasLabel("Label1")`, `hasLabel("Label2")` e `hasLabel("Label3")`.

> **Important**
>
> O delimitador `::` é reservado somente para esse uso. Você não pode especificar vários rótulos na etapa `hasLabel`. Por exemplo, `hasLabel("Label1::Label2")` não corresponde a nada.

Caracteres de escape

O Neptune resolve todos os caracteres de escape, conforme descrito na seção Escaping Special Characters da documentação da linguagem Apache Groovy.

Limitações do Groovy

O Neptune não é compatível com comandos do Groovy que não começam com `g`. Isso inclui matemática (por exemplo: `1+1`), chamadas do sistema (por exemplo: `System.nanoTime()`) e definições das variáveis (por exemplo: `1+1`).

> **Important**
>
> O Neptune não é compatível com nomes de classes totalmente qualificados. Por exemplo, você deve usar `single` e não `org.apache.tinkerpop.gremlin.structure.VertexProperty.Cardinality.single` na solicitação do Groovy.

Serialização

O Neptune é compatível com as serializações a seguir com base no tipo MIME solicitado.

Tipo MIME	Serialização	Configuração

application/vnd.gremlin-v1.0+gryo	GryoMessageSerializerV1	ioRegistries: [org.apache.tinkerpop.gremlin.tinkergraph.structure.TinkerIoRegistryV1]
application/vnd.gremlin-v1.0+gryo-stringd	GryoMessageSerializerV1	serializeResultToString: true}
application/vnd.gremlin-v3.0+gryo	GryoMessageSerializerV3	ioRegistries: [org.apache.tinkerpop.gremlin.tinkergraph.structure.TinkerIoRegistryV3]
application/vnd.gremlin-v3.0+gryo-stringd	GryoMessageSerializerV3	serializeResultToString: true
application/vnd.gremlin-v1.0+json	GraphSONMessageSerializerGremlinV1	ioRegistries: [org.apache.tinkerpop.gremlin.tinkergraph.structure.TinkerIoRegistryV1]
application/vnd.gremlin-v2.0+json	GraphSONMessageSerializerV2 (só funciona com WebSockets)	ioRegistries: [org.apache.tinkerpop.gremlin.tinkergraph.structure.TinkerIoRegistryV2]
application/vnd.gremlin-v3.0+json	GraphSONMessageSerializerV3	

`application/json`	`GraphSONMessageSerializerV3`	`ioRegistries: [org.apache.tinkerpop.gremlin.tinkergraph.structure.TinkerIoRegistryV3]`
`application/vnd.graphbinary-v1.0`	`GraphBinaryMessageSerializerV1`	

Embora o Neptune ofereça suporte a esses diferentes tipos de serializadores, a orientação para seu uso é bastante direta. Se você estiver se conectando ao Neptune via HTTP, priorize o uso `application/vnd.gremlin-v3.0+json;types=false` de, pois os tipos incorporados na versão alternativa do GraphSon 3 dificultam o trabalho. Se você estiver usando TinkerPop drivers Apache, provavelmente não precisará fazer nenhuma escolha, pois usaria o padrão de`application/vnd.graphbinary-v1.0`. Os formatos restantes permanecem presentes por motivos legados.

> **Note**
>
> A tabela do serializador mostrada aqui se refere à nomenclatura a partir da versão 3.7.0. TinkerPop Se você quiser saber mais sobre essa mudança, consulte a documentação de TinkerPop atualização. O suporte à serialização Gryo foi descontinuado na versão 3.4.3 e foi oficialmente removido na versão 3.6.0. Se você estiver usando explicitamente o Gryo ou em uma versão de driver que o usa por padrão, você deve mudar para o driver GraphBinary ou atualizá-lo.

Etapas do Lambda

O Neptune não é compatível com as etapas do Lambda.

Métodos do Gremlin não compatíveis

O Neptune não é compatível com os seguintes métodos do Gremlin:

- `org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversal.progra`
- `org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversal.sideEf`

- `org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversal.from(o`
- `org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversal.to(org`

Por exemplo, a seguinte travessia não é permitida:
`g.V().addE('something').from(__.V().next()).to(__.V().next())`

> **Important**
>
> Isso se aplica somente a métodos onde você envia a consulta Gremlin como uma string de texto.

Etapas do Gremlin não compatíveis

O Neptune não é compatível com as seguintes etapas de Gremlin:

- A Etapa io( ) do Gremlin é compatível apenas parcialmente com o Neptune. Ela pode ser usada em um contexto de leitura, como em `g.io(`*`(url)`*`).read()`, mas não para escrever.

Atributos do grafo do Gremlin no Neptune

A implementação do Gremlin no Neptune não expõe o objeto `graph`. As tabelas a seguir listam os atributos do Gremlin e indicam se o Neptune é compatível ou não com eles.

Compatibilidade do Neptune com atributos do **`graph`**

Os atributos de grafo do Neptune, quando compatíveis, são os mesmos que seriam gerados pelo comando `graph.features()`.

Atributo do grafo	Habilitado?
`Transactions`	verdadeiro
`ThreadedTransactions`	false
`Computer`	false
`Persistence`	verdadeiro

`ConcurrentAccess`	verdadeiro

Compatibilidade do Neptune com atributos de variável

Atributo de variável	Habilitado?
`Variables`	false
`SerializableValues`	false
`UniformListValues`	false
`BooleanArrayValues`	false
`DoubleArrayValues`	false
`IntegerArrayValues`	false
`StringArrayValues`	false
`BooleanValues`	false
`ByteValues`	false
`DoubleValues`	false
`FloatValues`	false
`IntegerValues`	false
`LongValues`	false
`MapValues`	false
`MixedListValues`	false
`StringValues`	false
`ByteArrayValues`	false
`FloatArrayValues`	false

`LongArrayValues`	false

Compatibilidade do Neptune com atributos do vértice

Atributo de vértice	Habilitado?
`MetaProperties`	false
`DuplicateMultiProperties`	false
`AddVertices`	verdadeiro
`RemoveVertices`	verdadeiro
`MultiProperties`	verdadeiro
`UserSuppliedIds`	verdadeiro
`AddProperty`	verdadeiro
`RemoveProperty`	verdadeiro
`NumericIds`	false
`StringIds`	verdadeiro
`UuidIds`	false
`CustomIds`	false
`AnyIds`	false

Compatibilidade do Neptune com atributos de propriedade do vértice

Atributo de propriedade de vértice	Habilitado?
`UserSuppliedIds`	false
`AddProperty`	verdadeiro

RemoveProperty	verdadeiro
NumericIds	verdadeiro
StringIds	verdadeiro
UuidIds	false
CustomIds	false
AnyIds	false
Properties	verdadeiro
SerializableValues	false
UniformListValues	false
BooleanArrayValues	false
DoubleArrayValues	false
IntegerArrayValues	false
StringArrayValues	false
BooleanValues	verdadeiro
ByteValues	verdadeiro
DoubleValues	verdadeiro
FloatValues	verdadeiro
IntegerValues	verdadeiro
LongValues	verdadeiro
MapValues	false
MixedListValues	false
StringValues	verdadeiro

`ByteArrayValues`	false
`FloatArrayValues`	false
`LongArrayValues`	false

Compatibilidade do Neptune com atributos de borda

Atributo de borda	Habilitado?
`AddEdges`	verdadeiro
`RemoveEdges`	verdadeiro
`UserSuppliedIds`	verdadeiro
`AddProperty`	verdadeiro
`RemoveProperty`	verdadeiro
`NumericIds`	false
`StringIds`	verdadeiro
`UuidIds`	false
`CustomIds`	false
`AnyIds`	false

Compatibilidade do Neptune com atributos de propriedade de borda

Atributo de propriedade de borda	Habilitado?
`Properties`	verdadeiro
`SerializableValues`	false
`UniformListValues`	false

`BooleanArrayValues`	false
`DoubleArrayValues`	false
`IntegerArrayValues`	false
`StringArrayValues`	false
`BooleanValues`	verdadeiro
`ByteValues`	verdadeiro
`DoubleValues`	verdadeiro
`FloatValues`	verdadeiro
`IntegerValues`	verdadeiro
`LongValues`	verdadeiro
`MapValues`	false
`MixedListValues`	false
`StringValues`	verdadeiro
`ByteArrayValues`	false
`FloatArrayValues`	false
`LongArrayValues`	false

Conformidade com os padrões SPARQL no Amazon Neptune

Depois de listar os padrões SPARQL aplicáveis, as seções a seguir fornecem detalhes específicos sobre como a implementação do SPARQL no Neptune se estende ou diverge desses padrões.

Tópicos

- Gráfico padrão e gráficos nomeados do SPARQL
- Funções de construtor XPath SPARQL compatíveis com o Neptune
- IRI de base padrão para consultas e atualizações
- Valores xsd:dateTime no Neptune
- Tratar valores de ponto flutuante especiais do Neptune
- Limitação de valores de comprimento arbitrário do Neptune
- O Neptune estende a comparação “igual a” no SPARQL
- Tratar literais fora do intervalo no SPARQL no Neptune

O Amazon Neptune está em conformidade com os padrões a seguir na implementação da linguagem de consulta de grafos SPARQL.

Padrões aplicáveis do SPARQL

- O SPARQL é definido pela recomendação W3C SPARQL 1.1 Query Language de 21 de março de 2013.
- O protocolo de atualização e a linguagem de consulta do SPARQL são definidos pela especificação W3C SPARQL 1.1 Update.
- Para formatos numéricos, o SPARQL segue a especificação W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes, consistente com a especificação IEEE 754 (IEEE 754-2019 - IEEE Standard for Floating-Point Arithmetic. Para obter mais informações, consulte também a página IEEE 754 da Wikipédia). No entanto, os recursos que foram apresentados após a versão `IEEE 754-1985` não estão incluídos na especificação.

Prefixos de namespace padrão no SPARQL no Neptune

O Neptune define os prefixos a seguir por padrão para uso em consultas do SPARQL. Para obter mais informações, consulte Nomes prefixados na especificação do SPARQL.

- `rdf – http://www.w3.org/1999/02/22-rdf-syntax-ns#`
- `rdfs – http://www.w3.org/2000/01/rdf-schema#`
- `owl – http://www.w3.org/2002/07/owl#`
- `xsd – http://www.w3.org/2001/XMLSchema#`

Gráfico padrão e gráficos nomeados do SPARQL

O Amazon Neptune associa cada triplo a um grafo nomeado. O gráfico padrão é definido como a união de todos os gráficos nomeados.

Gráfico padrão para consultas

Se você enviar uma consulta do SPARQL sem especificar explicitamente um gráfico por meio da palavra-chave `GRAPH` ou construções como `FROM NAMED`, o Neptune sempre considerará todos os trios em sua instância de banco de dados. Por exemplo, a seguinte consulta gera todos os triplos de um endpoint do SPARQL no Neptune:

```
SELECT * WHERE { ?s ?p ?o }
```

Trios que aparecem em mais de um gráfico são retornados somente uma vez.

Para obter informações sobre a especificação de gráfico padrão, consulte a seção Dataset do RDF da especificação do SPARQL 1.1 Query Language.

Especificar o gráfico nomeado para carregamentos, inserções ou atualizações

Se você não especificar um grafo nomeado ao carregar, inserir ou atualizar triplos, o Neptune usará o grafo nomeado fallback definido pelo URI `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`.

Ao emitir uma solicitação `Load` do Neptune usando um formato baseado em triplos, é possível especificar o grafo nomeado a ser usado para todos os triplos com o parâmetro `parserConfiguration: namedGraphUri`. Para obter mais informações sobre a sintaxe do comando `Load`, consulte the section called “Comando Loader”.

> **Important**
>
> Se você não usar esse parâmetro e não especificar um gráfico nomeado, o URI de fallback será usado: `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`.

Esse gráfico nomeado de fallback também será usado se você carregar trios por meio de `SPARQL UPDATE` sem fornecer explicitamente um destino de gráfico nomeado.

Você pode usar o formato com base em quads, N-Quads, para especificar um gráfico nomeado para cada trio no banco de dados.

> **Note**
>
> O uso de N-Quads permite que você deixe o gráfico nomeado em branco. Nesse caso, `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph` é usado. Você pode substituir o gráfico padrão nomeado para N-Quads usando a opção de configuração do analisador `namedGraphUri`.

Funções de construtor XPath SPARQL compatíveis com o Neptune

O padrão SPARQL permite que os mecanismos SPARQL sejam compatíveis com um conjunto extensível de funções do construtor XPath. No momento, o Neptune é compatível com as seguintes funções de construtor, nas quais o prefixo `xsd` é definido como `http://www.w3.org/2001/XMLSchema#`:

- `xsd:boolean`
- `xsd:integer`
- `xsd:double`
- `xsd:float`
- `xsd:decimal`
- `xsd:long`
- `xsd:unsignedLong`

IRI de base padrão para consultas e atualizações

Como um cluster do Neptune tem vários endpoints diferentes, usar o URL de solicitação de uma consulta ou atualização como IRI de base pode gerar resultados inesperados na resolução de IRIs relativos.

A partir da versão 1.2.1.0 do mecanismo, o Neptune usará `http://aws.amazon.com/neptune/default/` como IRI base se um IRI de base explícito não fizer parte da solicitação.

Na seguinte solicitação, o IRI de base faz parte da solicitação:

```
BASE <http://example.org/default/>
INSERT DATA { <node1> <id> "n1" }
```

```
BASE <http://example.org/default/>
SELECT * { <node1> ?p ?o }
```

E o resultado seria:

```
?p                                                  ?o
http://example.org/default/id                       n1
```

No entanto, nesta solicitação, nenhum IRI de base está incluído:

```
INSERT DATA { <node1> <id> "n1" }

SELECT * { <node1> ?p ?o }
```

Nesse caso, o resultado seria:

```
?p                                                  ?o
http://aws.amazon.com/neptune/default/id            n1
```

Valores xsd:dateTime no Neptune

Por motivos de desempenho, o Neptune sempre armazena valores de data/hora como Tempo Universal Coordenado (UTC). Isso torna as comparações diretas muito eficientes.

Também significa que se você inserir um valor `dateTime` que determine um fuso horário específico, o Neptune converterá o valor em UTC e descartará as informações de fuso horário. Depois, quando o valor `dateTime` for recuperado mais tarde, ele será expresso em UTC, não no fuso horário original, que não poderá mais ser determinado.

Tratar valores de ponto flutuante especiais do Neptune

O Neptune trata os valores de ponto flutuante especiais no SPARQL da forma a seguir.

Tratamento de NaN em SPARQL no Neptune

No Neptune, o SPARQL pode aceitar um valor de `NaN` em uma consulta. Não há distinção entre valores de `NaN` de sinalização e silenciosos. O Neptune trata todos os valores `NaN` como silenciosos.

Semanticamente, não é possível realizar uma comparação com um `NaN`, pois nada é maior que, menor que, nem igual a um `NaN`. Isso significa que um valor de `NaN` em um lado de uma comparação, teoricamente, nunca corresponde a nada do outro lado.

No entanto, a especificação XSD trata dois valores `xsd:double` ou `xsd:float` `NaN` como iguais. O Neptune aplica isso ao filtro `IN`, para o operador igual em expressões de filtro e para a semântica de correspondência exata (com `NaN` na posição do objeto para um padrão triplo).

Tratar valores infinitos do SPARQL no Neptune

No Neptune, o SPARQL pode aceitar um valor de `INF` ou `-INF` em uma consulta. O `INF` compara com outros valores numéricos superiores e o `-INF` compara com outros valores numéricos inferiores.

Dois valores INF com sinais correspondentes são comparados como iguais, independentemente do tipo (por exemplo, um `-INF` flutuante compara como igual a um `-INF` duplo).

Claro, não é possível realizar uma comparação com um `NaN`, pois nada é maior que, menor que, nem igual a um `NaN`.

Tratar zero negativo do SPARQL no Neptune

O Neptune normaliza um valor de zero negativo para um zero sem sinal. Você pode usar valores de zero negativo em uma consulta, mas eles não são registrados dessa forma no banco de dados, pois são comparados como iguais a zeros sem sinal.

Limitação de valores de comprimento arbitrário do Neptune

O Neptune limita o tamanho do armazenamento de valores inteiros XSD, de ponto flutuante e decimais no SPARQL a 64 bits. O uso de valores maiores gera um erro `InvalidNumericDataException`.

O Neptune estende a comparação “igual a” no SPARQL

O padrão do SPARQL define uma lógica ternária para expressões de valor, nas quais uma expressão de valor pode ser avaliada como `true`, `false` ou `error`. A semântica padrão para a igualdade de termo conforme definido na especificação SPARQL 1.1), que se aplica às comparações de = e `!=` nas condições de `FILTER`, produz um `error` ao comparar os tipos de dados que não são explicitamente comparáveis na tabela de operadores na especificação.

Esse comportamento pode levar a resultados não intuitivos, conforme o exemplo a seguir.

Dados:

```
<http://example.com/Server/1> <http://example.com/ip> "127.0.0.1"^^<http://example.com/
datatype/IPAddress>
```

Consulta 1:

```
SELECT * WHERE {
    <http://example.com/Server/1> <http://example.com/ip> ?o .
    FILTER(?o = "127.0.0.2"^^<http://example.com/datatype/IPAddress>)
}
```

Consulta 2:

```
SELECT * WHERE {
    <http://example.com/Server/1> <http://example.com/ip> ?o .
    FILTER(?o != "127.0.0.2"^^<http://example.com/datatype/IPAddress>)
}
```

Com a semântica SPARQL padrão usada pelo Neptune antes da versão 1.0.2.1, as duas consultas gerariam o resultado vazio. O motivo é que `?o = "127.0.0.2"^^<http://example.com/IPAddress>` quando avaliado para `?o := "127.0.0.1"^^<http://example.com/IPAddress>` gera um `error` em vez de `false`, porque não há regras de comparação explícitas especificadas para o tipo de dados personalizado `<http://example.com/IPAddress>`. Como resultado, a versão negada na segunda consulta também gera um `error`. Em ambas as consultas, o `error` faz com que a solução candidata seja filtrada.

A partir da versão 1.0.2.1, o Neptune ampliou o operador de desigualdade do SPARQL de acordo com a especificação. Consulte a seção SPARQL 1.1 section on operator extensibility, que permite que os mecanismos definam regras adicionais sobre como comparar tipos de dados internos definidos pelos usuários e não comparáveis.

Ao usar essa opção, o Neptune trata uma comparação entre dois tipos de dados que não estão explicitamente definidos na tabela de mapeamento do operador como `true`, se os valores literais e os tipos de dados forem sintaticamente iguais, e falsos, caso não sejam. Um `error` não será produzido nesses casos.

Ao usar essas nova semânticas, a segunda consulta retornaria um `"127.0.0.1"^^<http://example.com/IPAddress>` em vez de um resultado vazio.

Tratar literais fora do intervalo no SPARQL no Neptune

A semântica XSD define cada tipo numérico com o espaço de valor, exceto `integer` e `decimal`. Essas definições limitam todos os tipos a um intervalo de valores. Por exemplo, um intervalo de `xsd:byte` é de -128 a +127, inclusivo. Qualquer valor fora desse intervalo é considerado inválido.

Se você tentar atribuir um valor literal fora do espaço de valor de um tipo (por exemplo, se tentar definir an `xsd:byte` como um valor literal de 999), Netuno aceitará o valor como está, sem arredondá-lo ou truncá-lo. out-of-range Mas ele não continua como um valor numérico, porque o tipo determinado não pode representá-lo.

Ou seja, o Neptune aceita `"999"^^xsd:byte` mesmo estando fora do intervalo de valores `xsd:byte` definido. No entanto, após o valor continuar no banco de dados, ele poderá ser usado somente na semântica de correspondência exata, em uma posição do objeto de um padrão triplo. Nenhum filtro de intervalo pode ser executado nele porque out-of-range os literais não são tratados como valores numéricos.

A especificação SPARQL 1.1 define operadores de intervalo no formato `numeric-operator-numeric`, `string-operator-string`, `literal-operator-literal`, etc. O Neptune não pode executar um operador de comparação de intervalo como `invalid-literal-operator-numeric-value`.

Conformidade com a especificação OpenCypher no Amazon Neptune

A versão do Amazon Neptune do openCypher geralmente é compatível com as cláusulas, a sintaxe, as expressões, as funções e os operadores definidos na especificação atual do openCypher, que é a Cypher Query Language Reference versão 9. As limitações e as diferenças na compatibilidade do Neptune com o openCypher estão descritas abaixo.

> **Note**
>
> A implementação atual do Neo4j do Cypher contém funcionalidades que não estão contidas na especificação do openCypher mencionada acima. Se você estiver migrando o código Cypher atual para o Neptune, consulte Compatibilidade do Neptune com o Neo4j e Reformular consultas do Cypher para serem executadas no openCypher no Neptune para obter mais informações.

Compatibilidade com cláusulas do openCypher no Neptune

O Neptune é compatível com as seguintes cláusulas, exceto conforme observado:

- `MATCH`: compatível, exceto *`shortestPath()`* e *`allShortestPaths()`* no momento.
- `OPTIONAL MATCH`

- *MANDATORY MATCH*: no momento, não é compatível no Neptune. No entanto, o Neptune é compatível com valores de ID personalizados em consultas MATCH.
- RETURN: compatível, exceto quando usado com valores não estáticos para SKIP ou LIMIT. Por exemplo, o seguinte não funciona no momento:

```
MATCH (n)
RETURN n LIMIT toInteger(rand())    // Does NOT work!
```

- WITH: compatível, exceto quando usado com valores não estáticos para SKIP ou LIMIT. Por exemplo, o seguinte não funciona no momento:

```
MATCH (n)
WITH n SKIP toInteger(rand())
WITH count() AS count
RETURN count > 0 AS nonEmpty    // Does NOT work!
```

- UNWIND
- WHERE
- ORDER BY
- SKIP
- LIMIT
- CREATE: o Neptune permite criar valores de ID personalizados em consultas CREATE.
- DELETE
- SET
- REMOVE
- MERGE: o Neptune é compatível com valores de ID personalizados em consultas MERGE.
- *CALL[YIELD...]*: no momento, não é compatível no Neptune.
- UNION, UNION ALL: consultas somente leitura são compatíveis, mas consultas de mutação não são aceitas no momento.

Compatibilidade com operadores do openCypher no Neptune

O Neptune é compatível com os seguintes operadores, exceto conforme observado:

Operadores gerais

- `DISTINCT`
- O operador **`.`** para acessar as propriedades de um mapa literal aninhado.

Operadores matemáticos

- O operador de adição **`+`**.
- O operador de subtração **`-`**.
- O operador de multiplicação **`*`**.
- O operador de divisão **`/`**.
- O operador de divisão de módulo **`%`**.
- O operador de exponenciação **`^`** *`não é compatível`*.

Operadores de comparação

- O operador de adição **`=`**.
- O operador de desigualdade **`<>`**.
- O operador **`<`** menor que é compatível, exceto quando um dos argumentos é um caminho, uma lista ou um mapa.
- O operador **`>`** maior que é compatível, exceto quando um dos argumentos é um caminho, uma lista ou um mapa.
- O operador **`<=`** less-than-or-equal -to é suportado, exceto quando um dos argumentos é um caminho, uma lista ou um mapa.
- O operador **`>=`** greater-than-or-equal -to é suportado, exceto quando um dos argumentos é um caminho, uma lista ou um mapa.
- `IS NULL`
- `IS NOT NULL`
- `STARTS WITH` será compatível se os dados pesquisados forem uma string.
- `ENDS WITH` será compatível se os dados pesquisados forem uma string.
- `CONTAINS` será compatível se os dados pesquisados forem uma string.

Operadores booleanos

- `AND`
- `OR`
- `XOR`
- `NOT`

Operadores de string

- O operador de concatenação **+**.

Operadores List

- O operador de concatenação **+**.
- `IN` (confere a presença de um item em uma lista).

Compatibilidade com expressões do openCypher no Neptune

O Neptune é compatível com as seguintes expressões, exceto conforme observado:

- `CASE`
- No momento, a expressão **`[]`** não é compatível com o Neptune para acessar chaves de propriedade calculadas dinamicamente em um nó, um relacionamento ou um mapa. Por exemplo, o seguinte não funciona:

```
MATCH (n)
WITH [5, n, {key: 'value'}] AS list
RETURN list[1].name
```

Compatibilidade com funções do openCypher no Neptune

O Neptune é compatível com as seguintes funções, exceto conforme observado:

Funções de predicado

- `exists()`

Funções escalares

- `coalesce()`
- `endNode()`
- `epochmillis()`
- `head()`
- `id()`
- `last()`
- `length()`
- `randomUUID()`
- `properties()`
- `removeKeyFromMap`
- `size()`: esse método sobrecarregado atualmente só funciona para expressões de padrões, listas e strings.
- `startNode()`
- `timestamp()`
- `toBoolean()`
- `toFloat()`
- `toInteger()`
- `type()`

Agregar funções

- `avg()`
- `collect()`
- `count()`
- `max()`
- `min()`
- `percentileDisc()`
- `stDev()`
- `percentileCont()`
- `stDevP()`

- `sum()`

Listar as funções

- `join()` (concatena strings em uma lista em uma única string).
- `keys()`
- `labels()`
- `nodes()`
- `range()`
- `relationships()`
- `reverse()`
- `tail()`

Funções matemáticas: numéricas.

- `abs()`
- `ceil()`
- `floor()`
- `rand()`
- `round()`
- `sign()`

Funções matemáticas: logarítmicas.

- `e()`
- `exp()`
- `log()`
- `log10()`
- `sqrt()`

Funções matemáticas: trigonométricas.

- `acos()`

- `asin()`
- `atan()`
- `atan2()`
- `cos()`
- `cot()`
- `degrees()`
- `pi()`
- `radians()`
- `sin()`
- `tan()`

Funções de string

- `join()` (concatena strings em uma lista em uma única string).
- `left()`
- `lTrim()`
- `replace()`
- `reverse()`
- `right()`
- `rTrim()`
- `split()`
- `substring()`
- `toLower()`
- `toString()`
- `toUpper()`
- `trim()`

Funções definidas pelo usuário

No momento, as *`funções definidas pelo usuário`* não são aceitas no Neptune.

Detalhes da implementação do openCypher específicos do Neptune

As seções a seguir descrevem as maneiras pelas quais a implementação do openCypher no Neptune pode diferir ou ir além da especificação do openCypher

Avaliações de caminho de comprimento variável (VLP) no Neptune

As avaliações de caminho de comprimento variável (VLP) descobrem caminhos entre os nós no grafo. O comprimento do caminho pode ser irrestrito em uma consulta. Para evitar ciclos, a especificação do openCypher especifica que cada borda deve ser percorrida no máximo uma vez por solução.

Para VLPs, a implementação do Neptune se desvia da especificação do openCypher, pois só é compatível com valores constantes para filtros de igualdade de propriedades. Considere a seguinte consulta:

```
MATCH (x)-[:route*1..2 {dist:33, code:x.name}]->(y) return x,y
```

Como o valor do filtro de igualdade de propriedade `x.name` não é uma constante, essa consulta gera uma `UnsupportedOperationException` com a mensagem: `Property predicate over variable-length relationships with non-constant expression is not supported in this release.`

Suporte temporal na implementação do Neptune OpenCypher (banco de dados Neptune 1.3.1.0 e versões anteriores)

No momento, o Neptune oferece compatibilidade limitada para funções temporais no openCypher. Ele é compatível com o tipo de dados `DateTime` para tipos temporais.

A função `datetime()` pode ser usada para obter a data e a hora UTC atuais da seguinte forma:

```
RETURN  datetime() as res
```

Os valores de data e hora podem ser convertidos a partir de dados armazenados no Neptune da seguinte forma:

```
MATCH (n) RETURN datetime(n.createdDate)
```

Os valores de data e hora podem ser analisados a partir de strings em um formato de "data e Thora" em que a data e a hora são expressas em uma das formas compatíveis abaixo:

Formatos de data suportados

- `yyyy-MM-dd`
- `yyyyMMdd`
- `yyyy-MM`
- `yyyy-DDD`
- `yyyyDDD`
- `yyyy`

Formatos de hora compatíveis

- `HH:mm:ssZ`
- `HHmmssZ`
- `HH:mm:ssZ`
- `HH:mmZ`
- `HHmmZ`
- `HHZ`
- `HHmmss`
- `HH:mm:ss`
- `HH:mm`
- `HHmm`
- `HH`

Por exemplo: .

```
RETURN datetime('2022-01-01T00:01')      // or another example:
RETURN datetime('2022T0001')
```

Observe que todos os valores de data/hora no openCypher do Neptune são armazenados e recuperados como valores UTC.

O openCypher no Neptune usa um relógio `statement`, o que significa que o mesmo instante no tempo é usado durante toda a duração de uma consulta. Uma consulta diferente na mesma transação pode usar um instante diferente no tempo.

O Neptune não é compatível com o uso de uma função em uma chamada para `datetime()`. Por exemplo, o seguinte não funcionará:

```
CREATE (:n {date:datetime(tostring(2021))})  // ---> NOT ALLOWED!
```

O Neptune é compatível com a função `epochmillis()` que converte uma `datetime` em `epochmillis`. Por exemplo: .

```
MATCH (n) RETURN epochMillis(n.someDateTime)
1698972364782
```

No momento, o Neptune não é compatível com outras funções e operações em objetos `DateTime`, como adição e subtração.

Suporte temporal na implementação do Neptune OpenCypher (Neptune Analytics e Neptune Database 1.3.2.0 e versões posteriores)

A seguinte funcionalidade de data e hora OpenCypher se aplica ao Neptune Analytics. Como alternativa, você pode usar o parâmetro labmode `DatetimeMillisecond=enabled` para ativar a seguinte funcionalidade de data e hora na versão 1.3.2.0 e superior do Neptune Engine. Para obter mais detalhes sobre o uso dessa funcionalidade no modo de laboratório, consulteSuporte estendido de data e hora.

- Support para milissegundos. O literal de data e hora sempre será retornado com milissegundos, mesmo que milissegundos sejam 0. (O comportamento anterior era truncar milissegundos.)

```
CREATE (:event {time: datetime('2024-04-01T23:59:59Z')})

# Returning the date returns with 000 suffixed representing milliseconds
MATCH(n:event)
RETURN n.time as datetime

{
  "results" : [ {
    "n" : {
      "~id" : "0fe88f7f-a9d9-470a-bbf2-fd6dd5bf1a7d",
      "~entityType" : "node",
      "~labels" : [ "event" ],
      "~properties" : {
        "time" : "2024-04-01T23:59:59.000Z"
      }
```

```
    }
  } ]
}
```

- Support para chamar a função datetime () sobre propriedades armazenadas ou resultados intermediários. Por exemplo, as consultas a seguir não eram possíveis antes desse recurso.

  Datetime () sobre propriedades:

```
// Create node with property 'time' stored as string
CREATE (:event {time: '2024-04-01T23:59:59Z'})

// Match and return this property as datetime
MATCH(n:event)
RETURN datetime(n.time) as datetime
```

  Datetime () em relação aos resultados intermediários:

```
// Parse datetime from parameter
UNWIND $list as myDate
RETURN datetime(myDate) as d
```

- Agora também é possível salvar as propriedades de data e hora criadas nos casos mencionados acima.

  Salvando a data e hora da propriedade de string de uma propriedade para outra:

```
// Create node with property 'time' stored as string
CREATE (:event {time: '2024-04-01T23:59:59Z', name: 'crash'})

// Match and update the same property to datetime type
MATCH(n:event {name: 'crash'})
SET n.time = datetime(n.time)

// Match and update another node's property
MATCH(e:event {name: 'crash'})
MATCH(n:server {name: e.servername})
SET n.time = datetime(e.time)
```

  Crie nós em lote a partir de um parâmetro com uma propriedade datetime:

```
// Batch create from parameter
```

```
UNWIND $list as events
CREATE (n:crash) {time: datetime(events.time)}
// Parameter value
{
  "x":[
    {"time":"2024-01-01T23:59:29", "name":"crash1"},
    {"time":"2023-01-01T00:00:00Z", "name":"crash2"}
  ]
}
```

- Support para um subconjunto maior de formatos de data e hora ISO8601. Consulte abaixo.

Formatos com compatibilidade

O formato de um valor de data e hora é [Data] T [Hora] [Fuso horário], onde T é o separador. Se um fuso horário explícito não for fornecido, o UTC (Z) será considerado o padrão.

Fuso horário

Os formatos de fuso horário suportados são:

- +/- HH: mm
- +/- Hmm
- +/-HH

A presença de um fuso horário em uma string de data e hora é opcional. Caso a diferença de fuso horário seja 0, Z pode ser usado em vez do postfixo de fuso horário acima para indicar a hora UTC. O intervalo suportado de um fuso horário é de - 14:00 a + 14:00.

Data

Se nenhum fuso horário estiver presente ou se o fuso horário for UTC (Z), os formatos de data suportados serão os seguintes:

> **Note**
>
> DDD se refere a uma data ordinal, que representa um dia do ano de 001 a 365 (366 em anos bissextos). Por exemplo, 2024-002 representa 2 de janeiro de 2024.

- `yyyy-MM-dd`

- `yyyyMMdd`
- `yyyy-MM`
- `yyyyMM`
- `yyyy-DDD`
- `yyyyDDD`
- `yyyy`

Se um fuso horário diferente de Z for escolhido, os formatos de data suportados serão limitados ao seguinte:

- `yyyy-MM-dd`
- `yyyy-DDD`
- `yyyyDDD`

O intervalo suportado para datas é de 1400-01-01 a 9999-12-31.

Tempo

Se nenhum fuso horário estiver presente ou se o fuso horário for UTC (Z), os formatos de horário suportados são:

- `HH:mm:ss.SSS`
- `HH:mm:ss`
- `HHmmss.SSS`
- `HHmmss`
- `HH:mm`
- `HHmm`
- `HH`

Se um fuso horário diferente de Z for escolhido, os formatos de horário suportados serão limitados ao seguinte:

- `HH:mm:ss`
- `HH:mm:ss.SSS`

Diferenças na semântica da linguagem openCypher do Neptune

O Neptune representa IDs de nós e relacionamentos como strings em vez de números inteiros. O ID é igual ao ID fornecido pelo carregador de dados. Se houver um namespace para a coluna, o namespace mais o ID. Consequentemente, a função `id` gera uma string em vez de um número inteiro.

O tipo de dados `INTEGER` é limitado a 64 bits. Ao converter valores de ponto flutuante ou string maiores em um número inteiro usando a função `TOINTEGER`, valores negativos são truncados em `LLONG_MIN` e valores positivos são truncados em `LLONG_MAX`.

Por exemplo: .

```
RETURN TOINTEGER(2^100)
>  9223372036854775807

RETURN TOINTEGER(-1 * 2^100)
>  -9223372036854775808
```

A função **`join()`** específica do Neptune

O Neptune implementa uma função `join()` que não está presente na especificação do openCypher. Ela cria uma string literal a partir de uma lista de literais de string e um delimitador de string. Usa dois argumentos:

- O primeiro argumento é uma lista de literais de string.
- O segundo argumento é a string delimitadora, que pode ser composta por zero, um ou mais de um caractere.

Exemplo:

```
join(["abc", "def", "ghi"], ", ")    // Returns "abc, def, ghi"
```

A função **`removeKeyFromMap()`** específica do Neptune

O Neptune implementa uma função `removeKeyFromMap()` que não está presente na especificação do openCypher. Ela remove uma chave especificada de um mapa e gera o novo mapa resultante.

A função utiliza dois argumentos:

- O primeiro argumento é o mapa do qual remover a chave.
- O segundo argumento é a chave a ser removida do mapa.

A função `removeKeyFromMap()` é particularmente útil em situações em que você deseja definir valores para um nó ou um relacionamento desenrolando uma lista de mapas. Por exemplo: .

```
UNWIND [{`~id`: 'id1', name: 'john'}, {`~id`: 'id2', name: 'jim'}] as val
CREATE (n {`~id`: val.`~id`})
SET n = removeKeyFromMap(val, '~id')
```

Valores de ID personalizados para propriedades de nó e relacionamento

A partir da versão 1.2.0.2 do mecanismo, o Neptune estendeu a especificação do openCypher para que agora você possa especificar os valores `id` dos nós e dos relacionamentos nas cláusulas `CREATE`, `MERGE` e `MATCH`. Isso permite que você atribua strings fáceis de usar em vez de UUIDs gerados pelo sistema para identificar nós e relacionamentos.

> **Warning**
>
> Essa extensão da especificação do openCypher é incompatível com versões anteriores, porque agora `~id` é considerado um nome de propriedade reservado. Se você já estiver usando `~id` como propriedade em seus dados e consultas, precisará migrar a propriedade existente para uma nova chave de propriedade e remover a antiga. Consulte O que fazer se você estiver usando atualmente `~id` como propriedade.

Veja um exemplo que mostra como criar nós e relacionamentos com IDS personalizados:

```
CREATE (n {`~id`: 'fromNode', name: 'john'})
  -[:knows {`~id`: 'john-knows->jim', since: 2020}]
  ->(m {`~id`: 'toNode', name: 'jim'})
```

Se você tentar criar um ID personalizado que já esteja em uso, o Neptune gerará um erro `DuplicateDataException`.

Veja um exemplo do uso de um ID personalizado em uma cláusula `MATCH`:

```
MATCH (n {`~id`: 'id1'})
```

```
RETURN n
```

Veja um exemplo do uso de IDs personalizados em uma cláusula `MERGE`:

```
MATCH (n {name: 'john'}), (m {name: 'jim'})
MERGE (n)-[r {`~id`: 'john->jim'}]->(m)
RETURN r
```

O que fazer se você estiver usando atualmente **`~id`** como propriedade

Com a versão 1.2.0.2 do mecanismo, a chave `~id` nas cláusulas do openCypher agora é tratada como `id` e não como uma propriedade. Isso significa que, se você tiver uma propriedade denominada `~id`, o acesso a ela se tornará impossível.

Se você estiver usando uma propriedade `~id`, antes de realizar a atualização para a versão do mecanismo `1.2.0.2` ou superior, é necessário primeiro migrar a propriedade `~id` existente para uma nova chave de propriedade e depois remover a propriedade `~id`. Por exemplo, a consulta a seguir:

- Cria uma propriedade denominada “newId” para todos os nós,
- copia o valor da propriedade “~id” para a propriedade “newId”
- e remove a propriedade “~id” dos dados

```
MATCH (n)
WHERE exists(n.`~id`)
SET n.newId = n.`~id`
REMOVE n.`~id`
```

O mesmo precisa ser feito para qualquer relacionamento nos dados que tenha uma propriedade `~id`.

Você também precisará alterar todas as consultas que estiver usando que façam referência a uma propriedade `~id`. Por exemplo, esta consulta:

```
MATCH (n)
WHERE n.`~id` = 'some-value'
RETURN n
```

... mudaria para isto:

```
MATCH (n)
WHERE n.newId = 'some-value'
RETURN n
```

Outras diferenças entre o openCypher e o Cypher do Neptune

- O Neptune só é compatível com conexões TCP para o protocolo Bolt. WebSocketsconexões para Bolt não são suportadas.
- O openCypher do Neptune remove o espaço em branco conforme definido pelo Unicode nas funções `trim()`, `ltrim()` e `rtrim()`.
- No openCypher do Neptune, `tostring(`duplo`)` não muda automaticamente para a notação E para valores grandes do duplo.
- Embora CREATE do openCypher não crie propriedades de vários valores, elas podem existir em dados criados usando o Gremlin. Se o openCypher do Neptune encontrar uma propriedade de vários valores, um dos valores será escolhido arbitrariamente, criando um resultado não determinístico.

Modelo de dados de grafo do Neptune

A unidade básica de dados de grafo do Amazon Neptune é um elemento de quatro posições (quadrante), semelhante a um quadrante do Resource Description Framework (RDF). Veja as quatro posições de um quadrante do Neptune:

- `subject  (S)`
- `predicate  (P)`
- `object  (O)`
- `graph  (G)`

Cada quadrante é uma instrução que faz uma declaração sobre um ou mais recursos. Uma instrução pode confirmar a existência de um relacionamento entre dois recursos ou pode anexar uma propriedade (par de valor-chave) a um recurso. Você pode pensar no valor do predicado do quadrante geralmente como o verbo da instrução. Ele descreve o tipo de relacionamento ou de propriedade que está sendo definido. O objeto é o destino do relacionamento ou o valor da propriedade. Veja os exemplos a seguir:

- Uma relação entre dois vértices pode ser representada armazenando o identificador do vértice de origem na posição `S`, o identificador do vértice de destino na posição `0` e o rótulo da borda na posição `P`.
- Uma propriedade pode ser representada armazenando o identificador do elemento na posição `S`, a chave da propriedade na posição `P` e o valor da propriedade na posição `0`.

A posição do gráfico `G` é usada de forma diferente em pilhas distintas. Para dados do RDF no Neptune, a posição `G` contém um identificador de grafo nomeado. Para gráficos de propriedades no Gremlin, ele é usado para armazenar o valor do ID de ponto no caso de um ponto. Em todos os outros casos, o padrão é um valor fixo.

Um conjunto de instruções quad com identificadores de recursos compartilhados cria um gráfico.

Dicionário de valores voltados para o usuário

O Neptune não armazena a maioria dos valores voltados para o usuário diretamente nos vários índices que ele mantém. Em vez disso, ele os armazena separadamente em um dicionário e os substitui nos índices por identificadores de 8 bytes.

- Todos os valores voltados para o usuário que entrariam em índices `S`, `P` ou `G` são armazenados no dicionário dessa forma.
- No índice 0, os valores numéricos são armazenados diretamente no índice (embutido). Isso inclui valores `date` e `datetime` (representados como milissegundos da época).
- Todos os outros valores voltados para o usuário que entrariam no índice 0 são armazenados no dicionário e representados no índice por IDs.

O dicionário contém um mapeamento direto de valores voltados para o usuário para IDs de 8 bytes em um índice `value_to_id`.

Ele armazena o mapeamento reverso de IDs de 8 bytes para valores em um dos dois índices, dependendo do tamanho dos valores:

- Um índice `id_to_value` associa IDs a valores voltados para o usuário que são menores que 767 bytes após a codificação interna.
- Um índice `id_to_blob` associa IDs a valores maiores voltados para o usuário.

Como as declarações são indexadas no Neptune

Ao consultar um gráfico de quads, para cada posição do quad, você pode especificar ou não uma restrição de valor. A consulta retorna todos os quads correspondentes às restrições que você especificou.

O Neptune usa índices para resolver consultas. No artigo de 2005, Optimized Index Structures for Querying RDF from the Web (Estruturas de índice otimizadas para consultas de RDF na Web), Andreas Harth e Stefan Decker observaram que há 16 (2^4) padrões de acesso possíveis para as quatro posições quad. Você pode consultar todos os 16 padrões com eficiência sem precisar verificar e filtrar usando seis índices de instrução quad. Cada índice de instrução quad usa uma chave composta de quatro valores de posição concatenados em uma ordem diferente.

```
     Access Pattern                                      Index key order
-------------------------------------------------    ---------------
 1.  ????  (No constraints; returns every quad)           SPOG
 2.  SPOG  (Every position is constrained)                SPOG
 3.  SPO?  (S, P, and O are constrained; G is not)        SPOG
 4.  SP??  (S and P are constrained; O and G are not)     SPOG
```

```
 5.  S???  (S is constrained; P, O, and G are not)          SPOG
 6.  S??G  (S and G are constrained; P and O are not)       SPOG

 7.  ?POG  (P, O, and G are constrained; S is not)          POGS
 8.  ?PO?  (P and O are constrained; S and G are not)       POGS
 9.  ?P??  (P is constrained; S, O, and G are not)          POGS

10.  ?P?G  (P and G are constrained; S and O are not)       GPSO
11.  SP?G  (S, P, and G are constrained; O is not)          GPSO
12.  ???G  (G is constrained; S, P, and O are not)          GPSO

13.  S?OG  (S, O, and G are constrained; P is not)          OGSP
14.  ??OG  (O and G are constrained; S and P are not)       OGSP
15.  ??O?  (O is constrained; S, P, and G are not)          OGSP

16.  S?O?  (S and O are constrained; P and G are not)       OSGP
```

O Neptune cria e mantém apenas três desses seis índices por padrão:

- O `SPOG` – usa uma chave composta de `Subject + Predicate + Object + Graph`.
- O `POGS` – usa uma chave composta de `Predicate + Object + Graph + Subject`.
- O `GPSO` – usa uma chave composta de `Graph + Predicate + Subject + Object`.

Esses três índices lidam com muitos dos padrões de acesso mais comuns. A manutenção de apenas três índices de instrução completos, em vez de seis, reduz significativamente os recursos necessários para oferecer suporte ao acesso rápido sem varredura e filtragem. Por exemplo, o índice `SPOG`, permite pesquisa eficiente sempre que um prefixo das posições, como o vértice ou o identificador do vértice e da propriedade, está vinculado. O índice `POGS` permite acesso eficiente quando apenas o rótulo do ponto ou da propriedade armazenado na posição `P` é vinculado.

A API de nível inferior para localizar instruções assume um padrão de instrução em que algumas posições são conhecidas e o restante é deixado para descoberta por pesquisa de índice. Ao compor as posições conhecidas em um prefixo de chave de acordo com a ordem de chaves de índice de um dos índices de declaração, o Neptune executa uma verificação de intervalo para recuperar todas as declarações correspondentes às posições conhecidas.

No entanto, um dos índices de declaração que o Neptune não cria por padrão é um índice `OSGP` de percurso inverso, que pode coletar predicados nos objetos e nos assuntos. Em vez disso, o Neptune, por padrão, monitora predicados distintos em um índice separado que ele usa para fazer

uma verificação de união de `{all P x POGS}`. Quando trabalhar com o Gremlin, um predicado corresponde a uma propriedade ou um rótulo de ponto.

Se o número de predicados distintos em um grafo se tornar grande, a estratégia de acesso padrão do Neptune poderá se tornar ineficiente. No Gremlin, por exemplo, uma etapa `in()` em que nenhum rótulo de borda é fornecido, ou qualquer etapa que use `in()` internamente, como `both()` ou `drop()`, pode se tornar bastante ineficiente.

Habilitar a criação do índice OSGP usando o modo de laboratório

Se o modelo de dados criar um grande número de predicados distintos, o desempenho poderá se degradar e aumentar os custos operacionais. Isto pode ser melhorado drasticamente usando o modo de laboratório para habilitar o índice OSGP, além de três índices mantidos pelo Neptune por padrão.

> **Note**
>
> Esse atributo está disponível a partir da versão 1.0.1.0.200463.0 do mecanismo do Neptune.

A habilitação do índice OSGP pode ter algumas desvantagens:

- A taxa de inserção pode diminuir em até 23%.
- O armazenamento aumenta em até 20%.
- As consultas de leitura que tocam todos os índices igualmente (o que é bastante raro) podem aumentar as latências.

No entanto, em geral, vale a pena habilitar o índice OSGP para clusters de banco de dados com um grande número de predicados distintos. Pesquisas baseadas em objetos se tornam altamente eficientes (por exemplo, localizar todas as bordas de entrada de um vértice, ou todos os sujeitos conectados a um objeto específico) e, como resultado, remover vértices também se torna muito mais eficiente.

> **Important**
>
> É possível habilitar o índice OSGP somente em um cluster de banco de dados vazio, antes de carregar quaisquer dados nele.

Declarações do Gremlin no modelo de dados do Neptune

Os dados do grafo de propriedades do Gremlin são expressos no modelo SPOG usando três classes de declaração, a saber:

- Instruções de rótulo de vértice
- Instruções de borda
- Instruções de propriedade

Para obter uma explicação de como eles são usados nas consultas do Gremlin, consulte Noções básicas de como as consultas do Gremlin funcionam no Neptune.

O cache de pesquisa do Neptune pode acelerar as consultas de leitura

O Amazon Neptune implementa um cache de pesquisa que usa o SSD baseado em NVMe da instância R5d para melhorar o desempenho de leitura para consultas com pesquisas frequentes e repetitivas de valores de propriedades ou literais RDF. O cache de pesquisa armazena temporariamente esses valores no volume SSD NVMe, no qual eles podem ser acessados rapidamente.

Esse atributo está disponível a partir da Mecanismo do Amazon Neptune versão 1.0.4.2.R2 (01/06/2021).

As consultas de leitura que geram as propriedades de um grande número de vértices e bordas, ou de muitos triplos de RDF, poderão ter uma alta latência se os valores das propriedades ou literais precisarem ser recuperados dos volumes de armazenamento do cluster em vez da memória. Os exemplos incluem consultas de leitura de longa duração que geram um grande número de nomes completos de um grafo de identidade ou de endereços IP de um grafo de detecção de fraudes. À medida que o número de valores de propriedade ou literais RDF gerados pela consulta aumenta, a memória disponível diminui e a execução da consulta pode se degradar significativamente.

Casos de uso do cache de pesquisa do Neptune

O cache de pesquisa só ajuda quando suas consultas de leitura estão gerando as propriedades de um número muito grande de vértices e bordas ou de triplos de RDF.

Para otimizar o desempenho da consulta, o Amazon Neptune usa o tipo de instância R5d para criar um grande cache para esses valores de propriedade ou literais. Recuperá-los do cache é, então, muito mais rápido do que recuperá-los dos volumes de armazenamento do cluster.

Como regra, só vale a pena habilitar o cache de pesquisa se todas as três condições a seguir forem atendidas:

- Você tem observado um aumento na latência nas consultas de leitura.
- Você também está observando uma queda na `BufferCacheHitRatio` CloudWatch métrica ao executar consultas de leitura (consulteMonitorando Neptune usando a Amazon CloudWatch).
- Suas consultas de leitura estão gastando muito tempo materializando os valores de retorno antes de renderizar os resultados (veja o exemplo do perfil do Gremlin abaixo para saber como determinar quantos valores de propriedade estão sendo materializados para uma consulta).

> **Note**
>
> Esse atributo é útil somente no cenário específico descrito acima. Por exemplo, o cache de pesquisa não ajuda em nada nas consultas de agregação. A menos que você esteja executando consultas que se beneficiariam do cache de pesquisa, não há motivo para usar um tipo de instância R5d em vez de um tipo de instância R5 equivalente e mais barato.

Se você estiver usando o Gremlin, poderá avaliar os custos de materialização de uma consulta com o API `profile` do Gremlin. Em “Operações de índice”, ele mostra o número de termos materializados durante a execução:

```
Index Operations
Query execution:
    # of statement index ops: 3
    # of unique statement index ops: 3
    Duplication ratio: 1.0
    # of terms materialized: 5273
Serialization:
    # of statement index ops: 200
    # of unique statement index ops: 140
    Duplication ratio: 1.43
    # of terms materialized: 32693
```

O número de termos não numéricos que são materializados é diretamente proporcional ao número de pesquisas de termos que o Neptune precisa realizar.

Usar o cache de pesquisa

O cache de pesquisa só está disponível em um tipo de instância R5d, em que ele é ativado automaticamente por padrão. As instâncias R5d do Neptune têm as mesmas especificações que as instâncias R5, além de até 1,8 TB de armazenamento SSD local baseado em NVMe. Os caches de pesquisa são específicos da instância, e as workloads que se beneficiam podem ser direcionadas especificamente para instâncias R5d em um cluster do Neptune, enquanto outras workloads podem ser direcionadas para R5 ou outros tipos de instância.

Para usar o cache de pesquisa em uma instância do Neptune, basta atualizar essa instância para o tipo de instância R5d. Ao fazer isso, o Neptune define automaticamente o parâmetro do cluster de banco de dados neptune_lookup_cache como `'enabled'` e cria o cache de pesquisa nessa

instância específica. Depois, você pode usar a API Status de instância para confirmar que o cache foi habilitado.

Da mesma forma, para desativar o cache de pesquisa em uma instância específica, reduza a escala da instância verticalmente de um tipo R5d para um tipo R5 de instância equivalente.

Quando uma instância R5d é iniciada, o cache de pesquisa é habilitado e está no modo de inicialização a frio, o que significa que está vazio. O Neptune primeiro confere no cache de pesquisa os valores das propriedades ou literais de RDF enquanto processa as consultas e os adiciona se ainda não estiverem presentes. Isso aquece gradualmente o cache.

Quando você direciona as consultas de leitura que exigem pesquisas de valor de propriedade ou de literal de RDF para uma instância de leitor R5d, o desempenho de leitura se degrada um pouco enquanto o cache está se aquecendo. No entanto, quando o cache é aquecido, o desempenho de leitura acelera significativamente, e você também pode observar uma queda nos custos de E/S relacionados às pesquisas que atingem o cache em vez do armazenamento em cluster. A utilização da memória também melhora.

Se sua instância de gravador for uma R5d, ela aquecerá seu cache de pesquisa automaticamente em cada operação de gravação. Essa abordagem aumenta um pouco a latência para consultas de gravação, mas aquece o cache de pesquisa com maior eficiência. Então, se você direcionar as consultas de leitura que exigem pesquisas de valor de propriedade ou literal de RDF para a instância de gravador, você começará a obter um desempenho de leitura aprimorado imediatamente, pois os valores já terão sido armazenados em cache no local.

Além disso, se você estiver executando o carregador em massa em uma instância de gravador R5d, poderá notar que seu desempenho está um pouco degradado em virtude do cache.

Como o cache de pesquisa é específico de cada nó, a substituição do host redefine o cache como uma inicialização a frio.

Você pode desabilitar temporariamente o cache de pesquisa em todas as instâncias do cluster de banco de dados definindo o parâmetro do cluster de banco de dados neptune_lookup_cache como `'disabled'`. Em geral, no entanto, faz mais sentido desabilitar o cache em instâncias específicas, reduzindo a escala delas verticalmente de tipos de instância R5d para R5.

Semântica de transação no Neptune

O Amazon Neptune foi projetado para oferecer compatibilidade com workloads de processamento transacional online (OLTP) altamente simultâneas por meio de grafos de dados. A especificação W3C SPARQL Query Language para RDF e a documentação da Apache TinkerPop Gremlin Graph Traversal Language não definem a semântica da transação para processamento simultâneo de consultas. Como o suporte a ACID e as garantias de transação bem-definidas podem ser muito importantes, aplicamos semânticas rigorosas para ajudar a evitar anomalias de dados.

Esta seção define essas semânticas e ilustra como elas se aplicam a alguns casos de uso comuns no Neptune.

Tópicos

Definição de níveis de isolamento

O "I" em `ACID` significa isolamento. O grau de isolamento de uma transação determina quantas ou quão poucas outras transações simultâneas podem afetar os dados nos quais ela opera.

O SQL:1992 Standard criou um vocabulário para descrever níveis de isolamento. Ele define três tipos de interações (que ele chama de fenômenos) que podem ocorrer entre duas transações simultâneas, `Tx1` e `Tx2`:

- `Dirty read`: isso ocorre quando o `Tx1` modifica um item e o `Tx2` lê esse item antes de o `Tx1` ter confirmado a alteração. Portanto, se o `Tx1` nunca conseguir confirmar a alteração, ou revertê-la, o `Tx2` lerá um valor que nunca chegou ao banco de dados.
- `Non-repeatable read`: isso acontece quando o `Tx1` lê um item, o `Tx2` modifica ou exclui esse item e confirma a alteração e o `Tx1` tenta ler o item novamente. O `Tx1` agora lê um valor diferente do anterior ou descobre que o item não existe mais.
- `Phantom read`: isso acontece quando o `Tx1` lê um conjunto de itens que satisfazem um critério de pesquisa e, depois, o `Tx2` adiciona um novo item que atende ao critério de pesquisa, e o `Tx1` repete a pesquisa. O `Tx1` agora obtém um conjunto de itens diferente do anterior.

Cada um desses três tipos de interação pode provocar inconsistências nos dados resultantes em um banco de dados.

O SQL:1992 standard definiu quatro níveis de isolamento que têm garantias diferentes em termos dos três tipos de interação e das inconsistências que eles podem produzir. Em todos os quatro níveis, uma transação pode ser garantida para ser executada completamente ou não:

- `READ UNCOMMITTED`: permite todos os três tipos de interação (ou seja, leituras sujas, leituras não repetíveis e leituras fantasmas).
- `READ COMMITTED`: leituras sujas não são possíveis, mas leituras não repetíveis e fantasmas são.
- `REPEATABLE READ`: nem leituras sujas nem leituras não repetíveis são possíveis, mas leituras fantasmas ainda são.
- `SERIALIZABLE`: nenhum dos três tipos de fenômeno de interação pode ocorrer.

O controle de simultaneidade multiversão (MVCC — multiversion concurrency control) permite outro tipo de isolamento, ou seja, o isolamento SNAPSHOT . Isso garante que uma transação opere em um snapshot de dados como ele se encontra quando a transação começa e que nenhuma outra transação possa alterar esse snapshot.

Níveis de isolamento de transação no Neptune

O Amazon Neptune implementa diferentes níveis de isolamento de transação para consultas somente leitura e de mutação. As consultas SPARQL e Gremlin são classificadas como somente leitura ou de mutação com base nos seguintes critérios:

- No SPARQL, há uma distinção clara entre consultas de leitura (`SELECT`, `ASK`, `CONSTRUCT` e `DESCRIBE` conforme definido na especificação SPARQL 1.1 Query Language) e consultas de mutação (`INSERT` e `DELETE` conforme definido na especificação SPARQL 1.1 Update).

  Observe que o Neptune trata várias consultas de mutação enviadas juntas (por exemplo, em uma mensagem `POST`, separada por ponto e vírgula) como uma única transação. Elas são garantidas para obter sucesso ou falhar como uma unidade atômica e, no caso de falha, alterações parciais serão revertidas.
- No entanto, no Gremlin, o Neptune classifica uma consulta como uma somente leitura ou de mutação dependendo de ela conter ou não etapas de caminho de consulta, como `addE()`, `addV()`, `property()` ou `drop()` que manipula dados. Se a consulta contiver uma etapa de caminho desse tipo, ela será classificada e executada como uma consulta de mutação.

Também é possível usar sessões permanentes no Gremlin. Para ter mais informações, consulte Sessões baseadas em script do Gremlin. Nessas sessões, todas as consultas, incluindo consultas somente leitura, são executadas sob o mesmo isolamento das consultas de mutação no endpoint de gravador.

Usando sessões somente leitura Bolt no openCypher, todas as consultas, incluindo consultas somente leitura, são executadas sob o mesmo isolamento das consultas de mutação no endpoint de gravador.

Tópicos

- Isolamento de consulta somente leitura no Neptune
- Isolamento de consulta de mutação no Neptune
- Resolução de conflitos usando tempos limite de espera de bloqueio
- Bloqueios de intervalo e falsos conflitos

Isolamento de consulta somente leitura no Neptune

O Neptune avalia consultas somente leitura em semântica de isolamento de snapshot. Isso significa que uma consulta somente leitura opera logicamente em um snapshot consistente do banco de dados obtido quando a avaliação da consulta começa. O Neptune pode então garantir que nenhum dos seguintes fenômenos aconteça:

- `Dirty reads`: consultas somente leitura no Neptune nunca verão dados não confirmados de uma transação simultânea.
- `Non-repeatable reads`: uma transação somente leitura que lê os mesmos dados mais de uma vez sempre obterá os mesmos valores.
- `Phantom reads`: uma transação somente leitura nunca lerá os dados que foram adicionados após o início da transação.

Como o isolamento de snapshots é obtido usando o controle de simultaneidade multiversão (MVCC — multiversion concurrency control), as consultas somente leitura não precisam bloquear dados e, portanto, não bloqueiam consultas de mutação.

As réplicas de leitura só aceitam consultas somente leitura, portanto, todas as consultas em réplicas de leitura são executadas sob semântica de isolamento de `SNAPSHOT`.

A única consideração adicional ao consultar uma réplica de leitura é que pode haver um pequeno atraso de replicação entre o gravador e as réplicas de leitura. Isso significa que uma atualização feita no gravador pode demorar um pouco para ser propagada para a réplica de leitura na qual você está lendo. O tempo real de replicação depende da carga de gravação na instância primária. A arquitetura Neptune oferece suporte à replicação de baixa latência e o atraso de replicação é instrumentado em uma métrica da Amazon. CloudWatch

Ainda assim, devido ao nível de isolamento de `SNAPSHOT`, as consultas de leitura sempre veem um estado consistente do banco de dados, mesmo que não seja o mais recente.

Nos casos em que você precise de uma forte garantia de que uma consulta observa o resultado de uma atualização anterior, envie a consulta para o próprio endpoint gravador em vez de para uma réplica de leitura.

Isolamento de consulta de mutação no Neptune

As leituras feitas como parte de consultas de mutação são executadas sob o isolamento de transação `READ COMMITTED`, o que exclui a possibilidade de leituras contaminadas. Indo além das garantias normais fornecidas para o isolamento da transação `READ COMMITTED`, o Neptune fornece a forte garantia de que nem leituras `NON-REPEATABLE` nem `PHANTOM` possam acontecer.

Essas fortes garantias são obtidas bloqueando registros e intervalos de registros ao ler dados. Isso evita que transações simultâneas façam inserções ou exclusões em intervalos de índice após serem lidas, o que garante as leituras repetíveis.

> **Note**
>
> No entanto, uma transação de mutação simultânea `Tx2` pode começar após o início da transação de mutação de `Tx1` e pode confirmar uma alteração antes que o `Tx1` bloqueie os dados para lê-los. Nesse caso, o `Tx1` verá a alteração do `Tx2` como se o `Tx2` tivesse concluído antes do `Tx1` começar. Como isso se aplica apenas a alterações confirmadas, uma `dirty read` nunca poderá ocorrer.

Para entender o mecanismo de bloqueio que o Neptune usa para consultas de mutação, é útil primeiro entender os detalhes do Neptune Modelo de dados de grafo e Estratégia de indexação. O Neptune gerencia dados usando três índices, a saber, `SPOG`, `POGS` e `GPSO`.

Para obter leituras repetidas para o nível de transação `READ COMMITTED`, o Neptune usa bloqueios de intervalos no índice que está sendo usado. Por exemplo, se uma consulta de mutação ler todas

as propriedades e bordas de saída de um vértice chamado `person1`, o nó bloqueará todo o intervalo definido pelo prefixo `S=person1` no índice `SPOG` antes de ler os dados.

O mesmo mecanismo se aplica ao usar outros índices. Por exemplo, quando uma transação de mutação pesquisa todos os pares de vértices de origem-destino de um determinado rótulo de borda usando o índice `POGS`, o intervalo do rótulo de borda na posição `P` será bloqueado. Qualquer transação simultânea, independentemente de ser uma consulta somente leitura ou de mutação, ainda poderá executar leituras dentro do intervalo bloqueado. No entanto, qualquer mutação que envolva a inserção ou a exclusão de novos registros no intervalo de prefixos bloqueados exigirá um bloqueio exclusivo e será impedida.

Em outras palavras, quando um intervalo do índice tiver sido lido por uma transação de mutação, há uma forte garantia de que esse intervalo não será modificado por nenhuma transação simultânea até o final da transação de leitura. Isso garante que nenhuma `non-repeatable reads` ocorrerá.

Resolução de conflitos usando tempos limite de espera de bloqueio

Se uma segunda transação tentar modificar um registro em um intervalo que uma primeira transação bloqueou, o Neptune detectará o conflito imediatamente e bloqueará a segunda transação.

Se nenhum deadlock de dependência for detectado, o Neptune aplicará automaticamente um mecanismo de tempo limite de espera de bloqueio, no qual a transação bloqueada aguardará até sessenta segundos para que a transação que mantém o bloqueio seja concluída e libera o bloqueio.

- Se o tempo limite de espera de bloqueio expirar antes que o bloqueio seja liberado, a transação bloqueada será revertida.
- Se o bloqueio for liberado dentro do tempo limite de espera de bloqueio, a segunda transação será desbloqueada e poderá ser concluída com êxito sem a necessidade de repetição.

No entanto, se o Neptune detectar um deadlock de dependência entre as duas transações, a reconciliação automática do conflito não será possível. Nesse caso, o Neptune cancela e reverte imediatamente uma das duas transações sem iniciar um tempo limite de espera de bloqueio. O Neptune se esforça ao máximo para reverter a transação que tem o menor número de registros inseridos ou excluídos.

Bloqueios de intervalo e falsos conflitos

O Neptune obtém bloqueios de intervalo usando bloqueios de lacuna. Bloqueio de lacuna é um bloqueio em uma lacuna entre os registros do índice ou um bloqueio na lacuna antes do primeiro ou depois do último registro do índice.

O Neptune usa a chamada tabela de dicionário para associar valores de ID numéricos a literais de string específicos. Veja um exemplo de estado dessa tabela de dicionário do Netuno:

String	ID
tipo	1
default_graph	2
person_3	3
person_1	5
knows	6
person_2	7
idade	8
edge_1	9
lives_in	10
Nova York	11
Pessoa	12
Local	13
edge_2	14

As strings acima pertencem a um modelo de grafo de propriedades, mas os conceitos também se aplicam igualmente a todos os modelos de grafo de RDF.

O estado correspondente do índice SPOG (Subject-Predicate-Object_Graph) é mostrado abaixo, à esquerda. À direita, as strings correspondentes são mostradas para ajudar a entender o que significam os dados do índice.

S (ID)	P (ID)	O (ID)	G (ID)		S (string)	P (string)	O (string)	G (string)
3	1	12	2		person_3	tipo	Pessoa	default_graph
5	1	12	2		person_1	tipo	Pessoa	default_graph
5	6	3	9		person_1	knows	person_3	edge_1
5	8	40	2		person_1	idade	40	default_graph
5	10	11	14		person_1	lives_in	Nova York	edge_2
7	1	12	2		person_2	tipo	Pessoa	default_graph
11	1	13	2		Nova York	tipo	Local	default_graph

Agora, se uma consulta de mutação ler todas as propriedades e bordas de saída de um vértice chamado `person_1`, o nó bloqueará todo o intervalo definido pelo prefixo `S=person_1` no índice SPOG antes de ler os dados. O bloqueio de intervalo colocaria bloqueios de lacuna em todos os registros correspondentes e no primeiro registro que não correspondesse. Os registros correspondentes seriam bloqueados, mas não os não correspondentes. O Neptune colocaria os bloqueios de lacuna da seguinte forma:

- `5 1 12 2` (lacuna 1)
- `5 6 3 9` (lacuna 2)
- `5 8 40 2` (lacuna 3)

- `5 10 11 14` (lacuna 4)
- `7 1 12 2` (lacuna 5)

Isso bloqueia os seguintes registros:

- `5 1 12 2`
- `5 6 3 9`
- `5 8 40 2`
- `5 10 11 14`

Nesse estado, as seguintes operações estão legitimamente bloqueadas:

- Inserção de uma nova propriedade ou borda para `S=person_1`. Uma nova propriedade diferente de `type` ou uma nova borda precisaria entrar nas lacunas 2, 3, 4 ou 5, todas bloqueadas.
- Exclusão de qualquer um dos registros existentes.

Ao mesmo tempo, algumas operações simultâneas seriam bloqueadas falsamente (gerando falsos conflitos):

- Todas as inserções de borda ou propriedade para `S=person_3` são bloqueadas porque precisariam entrar na lacuna 1.
- Todas as novas inserções de vértice às quais seja atribuído um ID entre 3 e 5 seriam bloqueadas porque teriam que entrar na lacuna 1.
- Todas as novas inserções de vértice às quais seja atribuído um ID entre 5 e 7 seriam bloqueadas porque teriam que entrar na lacuna 5.

Os bloqueios de lacuna não são precisos o suficiente para bloquear a lacuna de um predicado específico (por exemplo, para bloquear a lacuna 5 para o predicado `S=5`).

Os bloqueios de intervalo são colocados apenas no índice em que a leitura acontece. No caso acima, os registros são bloqueados somente no índice SPOG, não no POGS nem no GPSO. As leituras de uma consulta podem ser realizadas em todos os índices, dependendo dos padrões de acesso, que podem ser listados usando as APIs `explain` (para Sparql e Gremlin).

> **Note**
>
> Bloqueios de lacunas também podem ser usados para atualizações simultâneas seguras nos índices subjacentes, o que também pode gerar falsos conflitos. Esses bloqueios de lacuna são colocados independentemente do nível de isolamento ou das operações de leitura realizadas pela transação.

Falsos conflitos podem ocorrer não apenas quando transações simultâneas colidem devido a bloqueios de lacuna, mas também em alguns casos quando uma transação está sendo repetida após qualquer tipo de falha. Se a reversão acionada pela falha ainda estiver em andamento e os bloqueios anteriormente utilizados para a transação ainda não tiverem sido totalmente liberados, a nova tentativa encontrará um falso conflito e falhará.

Sob uma carga alta, você normalmente pode descobrir que de 3% a 4% das consultas de gravação falham em virtude de falsos conflitos. Para um cliente externo, esses conflitos falsos são difíceis de prever e devem ser tratados com novas tentativas.

Exemplos de semântica de transação do Neptune

Os exemplos a seguir ilustram diferentes casos de uso para a semântica de transação no Amazon Neptune.

Tópicos

Exemplo 1: Inserir uma propriedade somente se ela não existir

Suponha que você queira garantir que uma propriedade seja definida apenas uma vez. Por exemplo, suponha que várias consultas estejam tentando atribuir uma pontuação de crédito a uma pessoa simultaneamente. Você só quer que uma instância da propriedade seja inserida, e as outras consultas falhem porque a propriedade já foi definida.

```
# GREMLIN:
g.V('person1').hasLabel('Person').coalesce(has('creditScore'), property('creditScore',
 'AAA+'))

# SPARQL:
INSERT { :person1 :creditScore "AAA+" .}
WHERE  { :person1 rdf:type :Person .
         FILTER NOT EXISTS { :person1 :creditScore ?o .} }
```

A etapa `property()` do Gremlin insere uma propriedade com a chave e o valor fornecidos. A etapa `coalesce()` executa o primeiro argumento na primeira etapa e, se falhar, executa a segunda etapa:

Antes de inserir o valor da propriedade `creditScore` para um determinado vértice `person1`, uma transação deve tentar ler o valor de `creditScore` possivelmente inexistente de `person1`. Essa tentativa de leitura bloqueia o intervalo SP para `S=person1` e `P=creditScore` no índice SPOG em que o valor de `creditScore` existe ou será gravado.

Usar esse bloqueio de intervalo impede que qualquer transação simultânea insira um valor de `creditScore` simultaneamente. Quando há várias transações paralelas, no máximo uma delas pode atualizar o valor de cada vez. Isso exclui a anomalia de mais de uma propriedade `creditScore` ser criada.

Exemplo 2: Afirmar que um valor de propriedade é globalmente exclusivo

Suponha que você queira inserir uma pessoa com um número de seguro social como uma chave primária. Você quer que sua consulta de mutação garanta que, em nível global, ninguém mais no banco de dados tenha o mesmo número de seguro social:

```
# GREMLIN:
g.V().has('ssn', 123456789).fold()
  .coalesce(__.unfold(),
            __.addV('Person').property('name', 'John Doe').property('ssn', 123456789'))

# SPARQL:
INSERT { :person1 rdf:type :Person .
         :person1 :name "John Doe" .
         :person1 :ssn 123456789 .}
WHERE  { FILTER NOT EXISTS { ?person :ssn 123456789 } }
```

Este exemplo é semelhante ao anterior. A principal diferença é que o bloqueio do intervalo é feito no índice POGS em vez de no índice SPOG.

A transação que executa a consulta deve ler o padrão, `?person :ssn 123456789`, no qual as posições P e 0 estão vinculadas. O bloqueio do intervalo é feito no índice `POGS` para `P=ssn` e `0=123456789`.

- Se o padrão existir, nenhuma ação será executada.
- Se não existir, o bloqueio impedirá que qualquer transação simultânea insira esse número de seguro social também

Exemplo 3: Alterar uma propriedade se outra propriedade tiver um valor especificado

Suponha que vários eventos em um jogo movam uma pessoa do nível um para o nível dois e atribuam a ela uma nova propriedade `level2Score` definida como zero. Você precisa ter certeza de que várias instâncias simultâneas dessa transação não possam criar várias instâncias da propriedade de pontuação de nível dois. As consultas no Gremlin e no SPARQL podem ter a seguinte aparência.

```
# GREMLIN:
g.V('person1').hasLabel('Person').has('level', 1)
 .property('level2Score', 0)
 .property(Cardinality.single, 'level', 2)

# SPARQL:
DELETE { :person1 :level 1 .}
INSERT { :person1 :level2Score 0 .
         :person1 :level 2 .}
WHERE  { :person1 rdf:type :Person .
         :person1 :level 1 .}
```

No Gremlin, quando `Cardinality.single` é especificado, a etapa `property()` adiciona uma nova propriedade ou substitui um valor de propriedade existente pelo novo valor especificado.

Qualquer atualização em um valor de propriedade, como aumentar o `level` de 1 para 2, é implementada como uma exclusão do registro atual e a inserção de um novo registro com o novo valor de propriedade. Nesse caso, o registro com o nível de número 1 é excluído e um registro com o nível de número 2 é reinserido.

Para que a transação possa adicionar `level2Score` e atualizar o `level` de 1 para 2, ela deve primeiro validar se o valor de `level` atualmente é igual a 1. Ao fazer isso, ele usa um bloqueio de intervalo no prefixo `SPO` para `S=person1`, `P=level` e `O=1` no índice `SPOG`. Esse bloqueio impede

que transações simultâneas excluam o triplo da versão 1 e, como resultado, nenhuma atualização simultânea conflitante pode ocorrer.

Exemplo 4: Substituir uma propriedade existente

Alguns eventos podem atualizar a pontuação de crédito de uma pessoa para um novo valor (aqui BBB). Mas você quer ter certeza de que eventos simultâneos desse tipo não possam criar várias propriedades de pontuação de crédito para uma pessoa.

```
# GREMLIN:
g.V('person1').hasLabel('Person')
 .sideEffect(properties('creditScore').drop())
 .property('creditScore', 'BBB')

# SPARQL:
DELETE { :person1 :creditScore ?o .}
INSERT { :person1 :creditScore "BBB" .}
WHERE  { :person1 rdf:type :Person .
         :person1 :creditScore ?o .}
```

Esse caso é semelhante ao exemplo 3, exceto que, em vez de bloquear o prefixo SPO, o Neptune bloqueia o prefixo SP apenas com `S=person1` e `P=creditScore`. Isso evita que transações simultâneas insiram ou excluam triplos com a propriedade `creditScore` para o assunto `person1`.

Exemplo 5: Evitar propriedades ou bordas pendentes

A atualização em uma entidade não deve deixar um elemento pendente, ou seja, uma propriedade ou borda associada a uma entidade que não é tipada. Isso é um problema apenas no SPARQL, porque o Gremlin tem restrições integradas para evitar elementos pendentes.

```
# SPARQL:
tx1: INSERT { :person1 :age 23 } WHERE { :person1 rdf:type :Person }
tx2: DELETE { :person1 ?p ?o }
```

A consulta `INSERT` deve ler e bloquear o prefixo SPO com `S=person1`, `P=rdf:type` e `O=Person` no índice SPOG. O bloqueio impede que a consulta `DELETE` seja bem-sucedida em paralelo.

Na corrida entre a tentativa da consulta `DELETE` excluir o registro `:person1 rdf:type :Person` e a consulta `INSERT` ler o registro e criar um bloqueio de intervalo em seu SPO no índice SPOG, os seguintes resultados são possíveis:

- Se a consulta `INSERT` for confirmada antes da consulta `DELETE` ler e excluir todos os registros de `:person1`, `:person1` será removido completamente do banco de dados, incluindo o registro recém-inserido.
- Se a consulta `DELETE` for confirmada antes da consulta `INSERT` tentar ler o registro `:person1 rdf:type :Person`, a leitura observará a alteração confirmada. Ou seja, ela não encontra nenhum registro `:person1 rdf:type :Person` e, portanto, torna-se um no-op.
- Se a consulta `INSERT` ler antes da consulta `DELETE`, o triplo `:person1 rdf:type :Person` será bloqueado e a consulta `DELETE` será bloqueada até que a consulta INSERT seja confirmada, como no primeiro caso anterior.
- Se a `DELETE` ler antes da consulta `INSERT`, e a consulta `INSERT` tentar ler e bloquear o prefixo `SP0` do registro, será detectado um conflito. Isso ocorre porque o triplo foi marcado para remoção, e a consulta `INSERT` falha.

Em todas essas diferentes sequências possíveis de eventos, nenhuma borda pendente é criada.

Tratamento de exceções e novas tentativas

Quando as transações são canceladas devido a conflitos não resolvidos ou a tempos limite de espera de bloqueio, o Amazon Neptune responde com um `ConcurrentModificationException`. Para ter mais informações, consulte Códigos de erro do mecanismo. Como melhor prática, os clientes sempre devem capturar e lidar com essas exceções.

Em muitos casos, quando o número de instâncias `ConcurrentModificationException` é baixo, um mecanismo de repetição baseado em recuo exponencial funciona bem como uma maneira de lidar com elas. Nessa abordagem de repetição, o número máximo de novas tentativas e de tempo de espera geralmente depende do tamanho máximo e da duração das transações.

No entanto, se seu aplicativo tiver cargas de trabalho de atualização altamente simultâneas, e você observar um grande número de eventos `ConcurrentModificationException`, poderá modificar o aplicativo para reduzir o número de modificações simultâneas conflitantes.

Por exemplo, considere um aplicativo que faz atualizações frequentes em um conjunto de vértices e usa vários threads simultâneos para essas atualizações a fim de otimizar a taxa de transferência de gravação. Se cada thread executar continuamente consultas que atualizam uma ou mais propriedades de nó, as atualizações simultâneas do mesmo nó poderão produzir `ConcurrentModificationExceptions`. Isso, por sua vez, pode degradar o desempenho de gravação.

Você pode reduzir significativamente a probabilidade de tais colisões se puder serializar atualizações que provavelmente entrarão em conflito entre si. Por exemplo, se você puder garantir que todas as consultas de atualização de um determinado nó sejam feitas no mesmo thread (talvez usando uma atribuição baseada em hash), poderá ter certeza de que elas serão executadas uma após a outra em vez de simultaneamente. Embora ainda seja possível que um bloqueio de intervalo obtido em um nó vizinho possa causar uma `ConcurrentModificationException`, você elimina atualizações simultâneas no mesmo nó.

Clusters e instâncias de banco de dados do Amazon Neptune

Um cluster de banco de dados do Amazon Neptune gerencia o acesso aos dados por meio de consultas. Um cluster consiste em:

- Uma instância de banco de dados primária.
- Até 15 instâncias de banco de dados de réplica de leitura.

Todas as instâncias em um cluster compartilham o mesmo volume de armazenamento gerenciado subjacente, projetado para oferecer confiabilidade e alta disponibilidade.

Você se conecta às instâncias de banco de dados em seu cluster de banco de dados por meio dos endpoints do Neptune.

A instância de banco de dados primária em um cluster de banco de dados do Neptune

A instância de banco de dados principal coordena todas as operações de gravação no volume de armazenamento subjacente do cluster de banco de dados. Ela também é compatível com operações de leitura.

Só pode haver uma instância de banco de dados primária em um cluster de banco de dados do Neptune. Se a instância primária ficar indisponível, o Neptune automaticamente fará o failover para uma das instâncias de réplica de leitura com uma prioridade que você pode especificar.

Instâncias de banco de dados de réplica de leitura em um cluster de banco de dados do Neptune

Depois de criar a instância principal de um cluster de banco de dados, você poderá criar até 15 réplicas de leitura no cluster de banco de dados para oferecer compatibilidade com consultas somente leitura.

As instâncias de banco de dados de réplicas de leitura do Neptune funcionam bem para a escalabilidade de leitura porque são totalmente dedicadas a operações de leitura no volume de cluster. Todas as operações de gravação são gerenciadas pela instância principal. Cada instância de banco de dados de réplica de leitura tem o próprio endpoint.

Como o volume de armazenamento do cluster é compartilhado entre todas as instâncias em um cluster, todas as instâncias de réplica de leitura geram os mesmos dados para os resultados da

consulta com muito pouco atraso na replicação. Esse atraso é geralmente muito inferior a 100 milissegundos, depois que a instância primária grava uma atualização, embora ele possa ser um pouco maior quando o volume de operações de gravação é muito grande.

Ter uma ou mais instâncias de réplica de leitura disponíveis em diferentes zonas de disponibilidade pode aumentar a disponibilidade, pois as réplicas de leitura servem como destinos de failover para a instância primária. Ou seja, se a instância principal falhar, o Neptune promoverá uma instância de réplica de leitura para se tornar a instância principal. Quando isso acontece, há uma breve interrupção enquanto a instância promovida é reinicializada, durante a qual as solicitações de leitura e gravação feitas na instância principal falham com uma exceção.

Por outro lado, se o cluster de banco de dados não incluir nenhuma instância de réplica de leitura, o cluster de banco de dados permanecerá indisponível quando a instância primária falhar até que seja recriada. Recriar a instância primária leva muito mais tempo do que promover uma réplica de leitura.

Para garantir a alta disponibilidade, recomendamos criar uma ou mais instâncias de réplica de leitura que tenham a mesma classe de instância de banco de dados da instância primária e estejam localizadas em zonas de disponibilidade diferentes da instância primária. Consulte Tolerância a falhas para um cluster de banco de dados do Neptune.

Usando o console, você pode criar uma implantação Multi-AZ. Basta especificar Multi-AZ ao criar um cluster de banco de dados. Caso um cluster de banco de dados esteja em uma única zona de disponibilidade, você poderá torná-lo um cluster de banco de dados Multi-AZ adicionando uma réplica do Neptune a uma zona de disponibilidade diferente.

> **Note**
>
> Não é possível criar uma instância de réplica de leitura criptografada para um cluster de banco de dados do Neptune não criptografado ou uma instância de réplica de leitura não criptografada para um cluster de banco de dados do Neptune criptografado.

Para obter detalhes sobre como criar uma instância de banco de dados de réplica de leitura do Neptune, consulte Criar uma instância de leitor do Neptune usando o console.

Dimensionar instâncias de banco de dados em um cluster de banco de dados do Neptune

Dimensione as instâncias no cluster de banco de dados do Neptune com base nos requisitos de CPU e memória. O número de vCPUs em uma instância determina o número de threads de consulta que processam as consultas de entrada. A quantidade de memória em uma instância determina o tamanho do cache do buffer, usado para armazenar cópias de páginas de dados obtidas do volume de armazenamento subjacente.

Cada instância de banco de dados do Neptune tem um número de threads de consulta igual a duas vezes o número de vCPUs nessa instância. Uma `r5.4xlarge`, por exemplo, com 16 vCPUs, tem 32 threads de consulta e, portanto, pode processar 32 consultas simultaneamente.

Consultas adicionais que chegam enquanto todos os threads de consulta estão ocupados são colocadas em uma fila do lado do servidor e processadas de modo FIFO à medida que os threads de consulta se tornam disponíveis. Essa fila do lado do servidor pode conter cerca de 8 mil solicitações pendentes. Quando estiver cheio, o Neptune responderá a solicitações adicionais com uma `ThrottlingException`. Você pode monitorar o número de solicitações pendentes com a `MainRequestQueuePendingRequests` CloudWatch métrica ou usando o endpoint de status da consulta Gremlin com o parâmetro. `includeWaiting`

Do ponto de vista do cliente, o tempo de execução da consulta inclui qualquer tempo gasto na fila, além do tempo gasto para realmente executar a consulta.

Uma carga de gravação simultânea sustentada que utiliza todos os threads de consulta na instância de banco de dados primária mostra preferencialmente 90% ou mais de utilização da CPU, o que indica que todos os threads de consulta no servidor estão ativamente engajados em realizar um trabalho útil. No entanto, a utilização real da CPU geralmente é um pouco menor, mesmo sob uma carga de gravação simultânea sustentada. Isso geralmente ocorre porque os threads de consulta aguardam a conclusão das operações de E/S no volume de armazenamento subjacente. O Neptune usa gravações de quórum que fazem seis cópias de seus dados em três zonas de disponibilidade, e quatro desses seis nós de armazenamento devem reconhecer uma gravação para que ela seja considerada durável. Enquanto um thread de consulta aguarda esse quórum do volume de armazenamento, ele fica paralisado, o que reduz a utilização da CPU.

Se você tem uma carga de gravação serial na qual está executando uma gravação após a outra e aguardando a conclusão da primeira antes de iniciar a próxima, você pode esperar que a utilização da CPU seja ainda menor. A quantidade exata será uma função do número de vCPUs e threads de

consulta (quanto mais threads de consulta, menos CPU geral por consulta), com alguma redução causada pela espera pela E/S.

Para obter mais informações sobre como dimensionar melhor as instâncias de banco de dados, consulte Escolher o tipo certo de instância de banco de dados do Neptune. Para conhecer os preços de cada tipo de instância, consulte a página Preços do Neptune.

Monitorar o desempenho da instâncias de banco de dados no Neptune

Você pode usar CloudWatch métricas no Neptune para monitorar o desempenho de suas instâncias de banco de dados e acompanhar a latência da consulta conforme observada pelo cliente. Consulte Usando CloudWatch para monitorar o desempenho da instância de banco de dados no Neptune.

Armazenamento, confiabilidade e disponibilidade do Amazon Neptune.

O Amazon Neptune usa uma arquitetura de armazenamento distribuído e compartilhado que é escalada automaticamente à medida que suas necessidades de armazenamento de banco de dados aumentam.

Os dados do Neptune são armazenados em um volume de cluster, que é um volume virtual único que usa unidades baseadas em SSD Non-Volatile Memory Express (NVMe). O volume de cluster consiste em uma coleção de blocos lógicos conhecidos como segmentos. Cada um desses segmentos recebe 10 gigabytes (GB) de armazenamento. Os dados em cada segmento são replicados em seis cópias, que são então alocadas em três zonas de disponibilidade (AZs) na região AWS em que reside o cluster de banco de dados.

Quando um cluster de banco de dados do Neptune é criado, ele recebe um único segmento de 10 GB. À medida que o volume de dados aumenta e excede o armazenamento alocado no momento, o Neptune expande automaticamente o volume do cluster adicionando novos segmentos. Um volume de cluster Neptune pode crescer até um tamanho máximo de 128 tebibytes (TiB) em todas as regiões suportadas, exceto na China GovCloud e, onde está limitado a 64 TiB. No entanto, para versões de mecanismos anteriores a Versão: 1.0.2.2 (09/03/2020), o tamanho dos volumes do cluster é limitado a 64 TiB em todas as regiões.

O volume de cluster de banco de dados contém todos os dados do usuário, índices e dicionários (descritos na seção Modelo de dados de grafo do Neptune), bem como metadados internos, como logs de transações internas. Todos esses dados grafos, incluindo índices e logs internos, não podem exceder o tamanho máximo do volume do cluster.

Opção de armazenamento otimizada para E/S

O Neptune oferece dois modelos de preços para armazenamento:

- Armazenamento padrão: o armazenamento padrão fornece armazenamento econômico de banco de dados para aplicações com uso moderado a baixo de E/S.
- Armazenamento otimizado para E/S: com o armazenamento otimizado para E/S, você paga somente pelo armazenamento que está usando, a um custo maior do que o armazenamento padrão, e não paga nada pela E/S que usa.

O armazenamento otimizado para E/S foi desenvolvido para atender às necessidades de workloads de grafos de uso intensivo de E/S a um custo previsível, com baixa latência de E/S e throughput de E/S consistente.

Para obter mais informações, consulte Armazenamento otimizado para E/S.

Alocação de armazenamento do Neptune

Embora um volume de cluster do Neptune possa aumentar até 128 TiB (ou 64 TiB em algumas regiões), você só é cobrado pelo espaço realmente alocado. O espaço total alocado é determinado pela marca d'água alta de armazenamento, que é a quantidade máxima alocada ao volume do cluster a qualquer momento durante sua existência.

Isso significa que, mesmo que os dados do usuário sejam removidos de um volume de cluster, por exemplo, por meio de uma consulta de descarte como `g.V().drop()`, o espaço total alocado permanece o mesmo. O Neptune otimiza automaticamente o espaço alocado não utilizado para reutilização no futuro.

Além dos dados do usuário, dois tipos adicionais de conteúdo consomem espaço de armazenamento interno, ou seja, dados do dicionário e logs de transação internos. Embora os dados do dicionário sejam armazenados com dados do grafo, eles são persistidos indefinidamente, mesmo quando os dados do grafo compatíveis foram excluídos, o que significa que as entradas poderão ser reutilizadas se os dados forem reintroduzidos. Os dados de log internos são armazenados em um espaço de armazenamento interno separado que tem sua própria marca d'água alta. Quando um log interno expira, o armazenamento que ele ocupava pode ser reutilizado para outros logs, mas não para dados de grafo. A quantidade de espaço interno que foi alocada para registros está incluída no espaço total relatado pela `VolumeBytesUsed` CloudWatch métrica.

Confira Práticas recomendadas de armazenamento para conhecer maneiras de reduzir ao mínimo o armazenamento alocado e de reutilizar o espaço.

Faturamento de armazenamento do Neptune

Os custos de armazenamento são cobrados com base na marca d’água alta de armazenamento, conforme descrito acima. Embora seus dados sejam replicados em seis cópias, você só será cobrado por uma cópia dos dados.

Você pode determinar qual é o limite máximo de armazenamento atual do seu cluster de banco de dados monitorando a `VolumeBytesUsed` CloudWatch métrica (consulteMonitorando Neptune usando a Amazon CloudWatch).

Outros fatores que podem afetar seus custos de armazenamento do Neptune incluem snapshots e backup do banco de dados, que são cobrados separadamente como armazenamento de backup e se baseiam nos custos de armazenamento do Neptune (consulte Métricas do CloudWatch que são úteis para gerenciar o armazenamento de backup do Neptune).

No entanto, se você criar um clone do seu banco de dados, o clone apontará para o mesmo volume de cluster que seu próprio cluster de banco de dados usa, portanto, não há cobrança adicional de armazenamento pelos dados originais. Alterações subsequentes no clone usam o copy-on-write protocolo e resultam em custos adicionais de armazenamento.

Para obter mais informações sobre preços do Neptune, consulte Preços do Amazon Neptune.

Práticas recomendadas de armazenamento do Neptune

Como determinados tipos de dados consomem armazenamento permanente no Neptune, use estas práticas recomendadas para evitar grandes picos no crescimento do armazenamento:

- Ao projetar seu modelo de dados do grafo, evite, o máximo possível, usar chaves de propriedade e valores voltados para o usuário que sejam temporários por natureza.
- Se você planeja fazer alterações em seu modelo de dados, não carregue dados em um cluster de banco de dados existente usando o novo modelo até limpar os dados desse cluster de banco de dados usando a API de redefinição rápida. Geralmente, a melhor estratégia é carregar dados que usam um novo modelo em um novo cluster de banco de dados.
- As transações que trabalham com grandes volumes de dados geram logs internos grandes de modo correspondente, o que pode aumentar permanentemente a marca d'água alta do espaço de log interno. Por exemplo, uma única transação que exclua todos os dados do cluster de banco de dados pode gerar um enorme log interno que exigiria a alocação de uma grande quantidade de armazenamento interno e, assim, reduziria permanentemente o espaço disponível para dados de grafo.

  Para evitar isso, divida as transações grandes em transações menores e reserve um tempo entre elas para que os logs internos associados tenham a chance de expirar e liberar seu armazenamento interno para reutilização pelos logs subsequentes.
- Para monitorar o crescimento do volume do cluster Neptune, você pode definir CloudWatch um alarme na métrica. `VolumeBytesUsed` CloudWatch Isso poderá ser particularmente útil se os

dados estiverem atingindo o tamanho máximo do volume do cluster. Para obter mais informações, consulte Usando CloudWatch alarmes da Amazon.

A única maneira de reduzir o espaço de armazenamento usado pelo seu cluster de banco de dados quando você tem uma grande quantidade de espaço alocado não utilizado é exportar todos os dados do seu grafo e depois recarregá-los em um novo cluster de banco de dados. Consulte o serviço e utilitário de exportação de dados do Neptune para ver uma maneira fácil de exportar dados de um cluster de banco de dados e o carregador em massa do Neptune para conhecer uma maneira fácil de importar dados de volta para o Neptune.

> **Note**
>
> Criar e restaurar um snapshot não reduz a quantidade de armazenamento alocada para seu cluster de banco de dados, porque um snapshot retém a imagem original do armazenamento subjacente do cluster. Se uma quantidade considerável do armazenamento alocado não estiver sendo usada, a única maneira de reduzir a quantidade de armazenamento alocado é exportar os dados do grafo e recarregá-los em um novo cluster de banco de dados.

Confiabilidade e alta disponibilidade do armazenamento do Neptune

O Amazon Neptune foi projetado para ser confiável, durável e tolerante a falhas.

O fato de seis cópias dos dados do Neptune serem mantidas em três zonas de disponibilidade (AZs) garante que o armazenamento dos dados seja altamente durável, com probabilidade muito baixa de perda de dados. Os dados são replicados automaticamente em todas as zonas de disponibilidade, independentemente de haver instâncias de bancos de dados nelas, e a quantidade de replicação independe do número de instâncias de banco de dados no cluster.

Isso significa que é possível adicionar uma réplica de leitura rapidamente, porque o Neptune não cria uma cópia dos dados do grafo. Em vez disso, a réplica de leitura se conecta ao volume de cluster que já contém os dados. Da mesma forma, remover uma réplica de leitura não remove nenhum dos dados subjacentes.

Você pode excluir o volume do cluster e seus dados somente depois de excluir todas as suas instâncias de banco de dados.

O Neptune também detecta automaticamente as falhas nos segmentos que compõem o volume do cluster. Quando uma cópia dos dados em um segmento é corrompida, o Neptune repara

imediatamente esse segmento, usando outras cópias dos dados dentro do mesmo segmento para garantir que os dados reparados estejam atualizados. Como resultado, o Neptune evita a perda de dados e reduz a necessidade de point-in-time realizar uma restauração para se recuperar de uma falha no disco.

Conectar-se a endpoints do Amazon Neptune

O Amazon Neptune usa um cluster de instâncias de banco de dados em vez de uma única instância. Cada conexão com o Neptune é processada por uma instância de banco de dados específica. Quando você se conecta a um cluster do Neptune, o nome do host e a porta especificados apontam para um processador intermediário chamado de endpoint. Endpoint é um URL que contém um endereço de host e uma porta. Os endpoints do Neptune usam conexões Transport Layer Security/ Secure Sockets Layer (TLS/SSL) criptografadas.

O Neptune usa o mecanismo de endpoint para abstrair essas conexões, para que você não precise codificar os nomes de host nem escrever a própria lógica para redirecionar conexões quando algumas instâncias de banco de dados não estão disponíveis.

Usando endpoints, é possível associar todas as conexões com a instância apropriada ou o grupo de instâncias, dependendo do caso de uso. Os endpoints personalizados permitem que você se conecte a subconjuntos de instâncias de banco de dados. Os seguintes endpoints estão disponíveis em um cluster de banco de dados do Neptune:

Endpoints de cluster do Neptune

Endpoint de cluster é um endpoint de um cluster de banco de dados do Neptune que se conecta à instância de banco de dados principal atual desse cluster de banco de dados. Cada cluster de banco de dados do Neptune tem um endpoint de cluster e uma instância de banco de dados principal.

O endpoint de cluster dá suporte a failover para conexões de leitura/gravação para o cluster de banco de dados. Use o endpoint do cluster para todas as operações de gravação no cluster de banco de dados, incluindo inserções, atualizações, exclusões e alterações do idioma de definição de dados (DDL). Você também pode usar o endpoint de cluster para operações de leitura, como consultas.

Se a instância de banco de dados principal atual de um cluster de banco de dados falhar, o Neptune fará failover automático para uma nova instância de banco de dados principal. Durante um failover, o cluster de banco de dados continua atendendo a solicitações de conexão para o endpoint de cluster pela nova instância de banco de dados primária, com interrupção mínima de serviço.

O exemplo a seguir ilustra um endpoint de cluster de um cluster de banco de dados do Neptune.

```
mydbcluster.cluster-123456789012.us-east-1.neptune.amazonaws.com:8182
```

Endpoints de leitor do Neptune

Endpoint de leitor é um endpoint para um cluster de banco de dados do Neptune que se conecta a uma das réplicas do Neptune disponíveis para esse cluster de banco de dados. Cada cluster de banco de dados do Neptune tem um endpoint de leitor. Se houver mais de uma réplica do Neptune, o endpoint de leitor direcionará cada solicitação de conexão para uma das réplicas do Neptune.

O endpoint de leitor fornece roteamento ida e volta para conexões somente leitura para o cluster de banco de dados. Use o endpoint do leitor para operações de leitura, como consultas .

Não é possível usar o endpoint do leitor para operações de gravação, a menos que você tenha um cluster de instância única (um cluster sem réplicas de leitura). Nesse caso e somente nesse caso, o leitor poderá ser usado para operações de gravação, bem como operações de leitura.

O roteamento ida e volta do endpoint de leitor funciona alterando o host para o qual a entrada DNS aponta. Sempre que resolve o DNS, você recebe um IP diferente, e conexões são abertas em relação a esses IPs. Depois que uma conexão é estabelecida, todas as solicitações para essa conexão são enviadas para o mesmo host. O cliente deve criar uma nova conexão e resolver o registro DNS novamente para conseguir uma conexão com a réplica de leitura potencialmente diferente.

> **Note**
>
> WebSockets as conexões geralmente são mantidas ativas por longos períodos. Para obter diferentes réplicas de leitura, faça o seguinte:
>
> - Verificar se o cliente resolve a entrada DNS sempre que ele se conecta.
> - Feche a conexão e reconecte.

Vários software cliente podem resolver o DNS de maneiras diferentes. Por exemplo, se resolver o DNS e, em seguida, usar o IP para cada conexão, o cliente direcionará todas as solicitações para um único host.

O armazenamento em cache do DNS para clientes ou proxies resolve o nome DNS para o mesmo endpoint do cache. Esse é um problema dos cenários de failover e de roteamento ida e volta.

> **Note**
>
> Desative todas as configurações de armazenamento em cache DNS para forçar sempre a resolução DNS.

O cluster de banco de dados distribui solicitações de conexão ao endpoint de leitor entre as réplicas disponíveis do Neptune. Se o cluster de banco de dados contiver apenas uma instância de banco de dados primária, o endpoint de leitor atenderá a solicitações de conexão da instância de banco de dados primária. Se uma réplica do Neptune for criada para esse cluster de banco de dados, o endpoint de leitor continuará atendendo a solicitações de conexão com o endpoint de leitor da nova réplica do Neptune com interrupção mínima no serviço.

O exemplo a seguir ilustra um endpoint de leitor de um cluster de banco de dados do Neptune.

```
mydbcluster.cluster-ro-123456789012.us-east-1.neptune.amazonaws.com:8182
```

Endpoints de instância do Neptune

Endpoint de instância é um endpoint para uma instância de banco de dados em um cluster de banco de dados do Neptune que se conecta a essa instância de banco de dados específica. Cada instância de banco de dados em um cluster de banco de dados, independentemente do tipo de instância, tem o próprio endpoint de instância exclusivo. Então, há um endpoint para a instância de banco de dados principal atual do cluster de banco de dados. Há também um endpoint da instância para cada uma das réplicas do Neptune no cluster de banco de dados.

O endpoint de instância oferece controle direto sobre as conexões do cluster de banco de dados, em cenários nos quais usar o endpoint de cluster ou o endpoint de leitor talvez não seja apropriado. Por exemplo, o aplicativo cliente pode exigir balanceamento de carga refinado com base no tipo de carga. Nesse caso, é possível configurar vários clientes para se conectarem a réplicas diferentes do Neptune em um cluster de banco de dados para distribuir workloads de leitura.

O exemplo a seguir ilustra um endpoint de uma instância de banco de dados em um cluster de banco de dados do Neptune.

```
mydbinstance.123456789012.us-east-1.neptune.amazonaws.com:8182
```

Endpoints personalizados do Neptune

Um endpoint personalizado para um cluster do Neptune representa um conjunto de instâncias de banco de dados escolhido. Quando você se conecta ao endpoint, o Neptune escolhe uma das instâncias no grupo para processar a conexão. Você define a quais instâncias esse endpoint se refere e determina a finalidade do endpoint.

Um cluster de banco de dados do Neptune não tem endpoints personalizados até você criar um, e é possível criar até cinco endpoints personalizados para cada cluster provisionado do Neptune.

O endpoint personalizado oferece conexões de banco de dados com carga balanceada baseadas em critérios que não sejam somente leitura ou com recurso de leitura/gravação das instâncias de banco de dados. Como a conexão pode ir para qualquer instância de banco de dados associada ao endpoint, garanta que todas as instâncias dentro desse grupo compartilhem algumas características de capacidade de desempenho e memória. Ao usar endpoints personalizados, você normalmente não usa o endpoint leitor nesse cluster.

Esse atributo se destina a usuários avançados com tipos personalizados de workloads em que não seja prático manter todas as réplicas do Neptune no cluster idêntico. Com endpoints personalizados, é possível ajustar a capacidade das instâncias de banco de dados usadas em cada conexão.

Por exemplo, se você definir vários endpoints personalizados que se conectem a grupos de instâncias com diferentes classes de instância, poderá direcionar usuários com diferentes necessidades de desempenho para os endpoints que melhor se ajustem aos casos de uso deles.

O exemplo a seguir ilustra um endpoint personalizado de uma instância de banco de dados em um cluster de banco de dados do Neptune:

```
myendpoint.cluster-custom-123456789012.us-east-1.neptune.amazonaws.com:8182
```

Consulte Trabalhar com endpoints personalizados Para mais informações.

Considerações sobre endpoint do Neptune

Pense nos seguintes problemas ao trabalhar com endpoints do Neptune:

- Antes de usar um endpoint de instância para se conectar a uma instância de banco de dados específica em um cluster de banco de dados, considere usar o endpoint do cluster ou do leitor para o cluster de banco de dados.

O endpoint de cluster e o endpoint de leitor dão suporte a cenários de alta disponibilidade. Se a instância de banco de dados principal de um cluster de banco de dados falhar, o Neptune fará failover automaticamente para uma nova instância de banco de dados principal. Ela faz isso promovendo uma réplica do Neptune existente para uma nova instância de banco de dados principal ou criando uma instância de banco de dados principal. Se ocorrer um failover, você poderá usar o endpoint do cluster para se reconectar à instância de banco de dados principal recém-promovida ou criada ou usar o endpoint de leitor para se reconectar a uma das outras réplicas do Neptune no cluster de banco de dados.

Se não adotar essa abordagem, você ainda poderá verificar se está se conectando à instância de banco de dados certa no cluster de banco de dados da operação desejada. Para isso, você pode descobrir de maneira manual ou programática o conjunto resultante de instâncias de banco de dados disponíveis no cluster de banco de dados e confirmar os tipos de instância após o failover, antes de usar o endpoint de instância de uma instância de banco de dados específica.

Para obter mais informações sobre failovers, consulte Tolerância a falhas para um cluster de banco de dados do Neptune.

- O endpoint de leitor só direciona conexões para réplicas do Neptune disponíveis em um cluster de banco de dados do Neptune. Ele não direciona consultas específicas.

  > **Important**
  >
  > O Neptune não faz balanceamento de carga.

  Se você deseja equilibrar a carga das consultas para distribuir a carga de trabalho de leitura para um cluster de banco de dados, precisará gerenciar isso no aplicativo. Você deve usar endpoints de instância para se conectar diretamente a réplicas do Neptune para equilibrar a carga.

- O roteamento ida e volta do endpoint de leitor funciona alterando o host para o qual a entrada DNS aponta. O cliente deve criar uma nova conexão e resolver o registro DNS novamente para conseguir uma conexão com a réplica de leitura potencialmente nova.

- Durante um failover, o endpoint de leitor pode direcionar conexões para a nova instância de banco de dados principal de um cluster de banco de dados por um curto período quando uma réplica do Neptune é promovida para a nova instância de banco de dados principal.

Trabalhar com endpoints personalizados no Neptune

Quando você adiciona uma instância de banco de dados a um endpoint personalizado ou a remove de um endpoint personalizado, todas as conexões existentes com essa instância de banco de dados permanecem ativas.

É possível definir uma lista de instâncias de banco de dados para serem incluídas em um endpoint personalizado (a lista estática) ou uma para excluir do endpoint personalizado (a lista de exclusões). Use o mecanismo de inclusão/exclusão para subdividir as instâncias de banco de dados em grupos e verifique se os endpoints personalizados abrangem todas as instâncias de banco de dados no cluster. Cada endpoint personalizado só pode conter um desses tipos de lista.

No AWS Management Console, a escolha é representada pela caixa de seleção Attach future instances added to this cluster. Quando você mantém a caixa de seleção desmarcada, o endpoint personalizado usa uma lista estática contendo apenas as instâncias de banco de dados especificadas na caixa de diálogo. Quando você marca a caixa de seleção, o endpoint personalizado usa uma lista de exclusões. Nesse caso, o endpoint personalizado representa todas as instâncias de banco de dados no cluster (inclusive as adicionadas futuramente), exceto as deixadas desmarcadas na caixa de diálogo.

O Neptune não altera as instâncias de banco de dados especificadas nas listas estáticas ou de exclusão quando as instâncias de banco de dados mudam de funções entre a instância principal e a réplica do Neptune em virtude do failover ou da promoção.

Associe uma instância de banco de dados a mais de um endpoint personalizado. Por exemplo, suponhamos que você adicione uma nova instância de banco de dados a um cluster. Nesse caso, a instância de banco de dados é adicionada a todos os endpoints personalizados para os quais está qualificada. A lista estática ou de exclusões definida para ela determina qual instância de banco de dados pode ser adicionada a ela.

Se o endpoint incluir uma lista estática de instâncias de banco de dados, as réplicas do Neptune recém-adicionadas não serão adicionadas a ele. Por outro lado, se o endpoint tiver uma lista de exclusões, as réplicas do Neptune recém-adicionadas serão adicionadas a ela, desde que não estejam indicadas na lista de exclusões.

Caso uma réplica do Neptune se torne indisponível, ela continuará associada aos seus endpoints personalizados. Isso será válido se ela não estiver íntegra, for interrompida, for reinicializada ou estiver indisponível por outro motivo. Porém, enquanto ela permanecer indisponível, não será possível se conectar a ela por meio de nenhum endpoint.

Como clusters do Neptune recém-criados não têm endpoints personalizados, é necessário criá-los e gerenciá-los por conta própria. Isso também vale para clusters do Neptune restaurados por meio de snapshots, porque os endpoints personalizados não estão incluídos no snapshot. Você os recriará após a restauração e escolherá novos nomes de endpoint se o cluster restaurado estiver na mesma região do original.

Criar um endpoint personalizado

Gerencie endpoints personalizados usando o console do Neptune. Faça isso acessando a página de detalhes do cluster do Neptune e use os controles na seção Endpoints personalizados.

1. Faça login no AWS Management Console e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. Acesse a página de detalhes do cluster.
3. Escolha a ação `Create custom endpoint` na seção Endpoints.
4. Selecione um nome para o endpoint personalizado, exclusivo para o ID do usuário e a região. O nome deve ter 63 caracteres ou menos e ter o seguinte formato:

   endpointName`.cluster-custom-`*customerDnsIdentifier*`.`*dnsSuffix*

   Como os nomes de endpoint personalizados não incluem o nome do cluster, você vai precisa alterar esses nomes se renomear um cluster. No entanto, não é possível reutilizar o mesmo nome de endpoint personalizado em mais de um cluster na mesma região. Atribua a cada endpoint personalizado um nome que seja exclusivo entre os clusters de propriedade do ID do usuário dentro de uma região específica.
5. Para escolher uma lista de instâncias de banco de dados que permaneça a mesma mesmo quando o cluster se expande, mantenha a caixa de seleção Attach future instances added to this cluster (Anexar instâncias futuras adicionadas a esse cluster) desmarcada. Quando essa caixa de seleção é marcada, o endpoint personalizado adiciona de maneira dinâmica todas as novas instâncias ao adicioná-las ao cluster.

Visualizar endpoints personalizados

1. Faça login no AWS Management Console e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. Acesse a página de detalhes do cluster de banco de dados.
3. A seção Endpoints contém apenas informações sobre endpoints personalizados (os detalhes sobre os endpoints integrados estão listados na seção Detalhes principal). Para ver os detalhes de um endpoint personalizado específico, selecione o nome para abrir a página de detalhes desse endpoint.

Editar um endpoint personalizado

É possível editar as propriedades de um endpoint personalizado para alterar quais instâncias de banco de dados estão associadas a ele. Você também pode alternar entre uma lista estática e uma lista de exclusões.

Não se conecte a ou use um endpoint personalizado enquanto as alterações em uma ação de edição estão em andamento. Pode demorar alguns minutos para o status do endpoint retornar a Disponível e você se reconectar.

1. Faça login no AWS Management Console e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. Acesse a página de detalhes do cluster.
3. Na seção Endpoints, selecione o nome do endpoint personalizado que você deseja editar.
4. Na página de detalhes desse endpoint, escolha a ação Editar.

Excluir um endpoint personalizado

1. Faça login no AWS Management Console e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. Acesse a página de detalhes do cluster.
3. Na seção Endpoints, selecione o nome do endpoint personalizado que você deseja excluir.
4. Na página de detalhes desse endpoint, escolha a ação Excluir.

Injetar um ID personalizado em uma consulta do Gremlin ou do SPARQL no Neptune

Por padrão, o Neptune atribui um valor exclusivo de `queryId` a cada consulta. Você pode usar esse ID para obter informações sobre uma consulta em execução (consulte API de status de consulta do Gremlin ou API de status de consulta do SPARQL) ou para cancelá-la (consulte Cancelamento de consultas do Gremlin ou Cancelamento de consulta do SPARQL).

O Neptune também permite que você especifique seu próprio valor de `queryId` para uma consulta do Gremlin ou do SPARQL, no cabeçalho HTTP ou para uma consulta do SPARQL usando a dica de consulta de `queryId`. A atribuição de seu próprio `queryID` facilita manter o controle de uma consulta para obter o status ou cancelá-la.

> **Note**
>
> Esse atributo está disponível a partir da Versão 1.0.1.0.200463.0 (15/10/2019).

Injetar um valor de **`queryId`** personalizado usando o cabeçalho HTTP

Para o Gremlin e o SPARQL, o cabeçalho HTTP pode ser usado para injetar seu próprio valor de `queryId` em uma consulta.

Exemplo do Gremlin

```
curl -XPOST https://your-neptune-endpoint:port \
    -d "{\"gremlin\": \
        \"g.V().limit(1).count()\" , \
        \"queryId\":\"4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47\"  }"
```

Exemplo do SPARQL

```
curl https://your-neptune-endpoint:port/sparql \
    -d "query=SELECT * WHERE { ?s ?p ?o } " \
       --data-urlencode \
       "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47"
```

Injetar um valor de **`queryId`** personalizado usando uma dica de consulta do SPARQL

Veja a seguir um exemplo de como usar a dica de consulta `queryId` do SPARQL para injetar um valor de `queryId` personalizado em uma consulta do SPARQL:

```
curl https://your-neptune-endpoint:port/sparql \
    -d "PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#> \
       SELECT * WHERE { hint:Query hint:queryId \"4d5c4fae-
aa30-41cf-9e1f-91e6b7dd6f47\" \
       {?s ?p ?o}}"
```

Usar o valor de **`queryId`** para verificar o status da consulta

Exemplo do Gremlin

```
curl https://your-neptune-endpoint:port/gremlin/status \
    -d "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47"
```

Exemplo do SPARQL

```
curl https://your-neptune-endpoint:port/sparql/status \
    -d "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47"
```

Modo de laboratório do Neptune

É possível usar o modo de laboratório do Amazon Neptune para habilitar novos atributos que estão na versão atual do mecanismo do Neptune, mas que ainda não estão prontos para uso em produção e não estão habilitados por padrão. Isso permite testar esses recursos em seus ambientes de desenvolvimento e teste.

> **Note**
>
> Esse atributo está disponível a partir da Versão 1.0.1.0.200463.0 (15/10/2019).

Usar o modo de laboratório do Neptune

Use o parâmetro de cluster de banco de dados `neptune_lab_mode` para habilitar ou desabilitar atributos. Para fazer isso, inclua *`(feature name)`*`=enabled` ou *`(feature name)`*`=disabled` no valor do parâmetro `neptune_lab_mode` no grupo de parâmetros do cluster de banco de dados.

Por exemplo, nesta versão do mecanismo, você pode definir o parâmetro `neptune_lab_mode` como `Streams=disabled, ReadWriteConflictDetection=enabled`.

Para obter informações sobre como editar o grupo de parâmetros do cluster de seu banco de dados, consulte Edição de um grupo de parâmetros. Observe que você não pode editar o grupo de parâmetros de cluster de banco de dados padrão. Se estiver usando o grupo padrão, você deverá criar um novo grupo de parâmetros de cluster de banco de dados antes de definir o parâmetro `neptune_lab_mode`.

> **Note**
>
> Ao fazer uma alteração em um parâmetro estático do cluster de banco de dados, como `neptune_lab_mode`, é necessário reiniciar a instância primária (de gravador) do cluster para que a alteração tenha efeito. Antes de Versão: 1.2.0.0 (21/07/2022), todas as réplicas de leitura em um cluster de banco de dados eram reinicializadas automaticamente quando a instância primária era reiniciada.
> A partir de Versão: 1.2.0.0 (21/07/2022), reiniciar a instância primária não faz com que as réplicas sejam reiniciadas. Isso significa que você deve reiniciar cada instância separadamente para obter uma alteração no parâmetro do cluster de banco de dados (consulte Grupos de parâmetros).

Important

No momento, se você fornecer os parâmetros incorretos do modo de laboratório ou se sua solicitação falhar por outro motivo, talvez você não seja notificado da falha. É necessário sempre verificar se uma solicitação de alteração no modo de laboratório foi bem-sucedida chamando a API de status, conforme mostrado abaixo:

```
curl -G https://your-neptune-endpoint:port/status
```

Os resultados do status incluem informações do modo de laboratório que mostrarão se as alterações solicitadas foram feitas ou não:

```
{
  "status":"healthy",
  "startTime":"Wed Dec 29 02:29:24 UTC 2021",
  "dbEngineVersion":"development",
  "role":"writer",
  "dfeQueryEngine":"viaQueryHint",
  "gremlin":{"version":"tinkerpop-3.5.2"},
  "sparql":{"version":"sparql-1.1"},
  "opencypher":{"version":"Neptune-9.0.20190305-1.0"},
  "labMode":{
    "ObjectIndex":"disabled",
    "ReadWriteConflictDetection":"enabled"
  },
  "features":{
    "LookupCache":{"status":"Available"},
    "ResultCache":{"status":"disabled"},
    "IAMAuthentication":"disabled",
    "Streams":"disabled",
    "AuditLog":"disabled"
  },
  "settings":{"clusterQueryTimeoutInMs":"120000"}
}
```

No momento, os seguintes atributos são acessados no modo de laboratório:

O índice OSGP

O Neptune agora pode manter um quarto índice, chamado de índice OSGP, que é útil para conjuntos de dados com um grande número de predicados (consulte Habilitar um índice OSGP).

> **Note**
>
> Esse atributo está disponível a partir da versão 1.0.2.1 do mecanismo do Neptune.

É possível habilitar um índice OSGP em um novo cluster de banco de dados vazio do Neptune definindo `ObjectIndex=enabled` no parâmetro do cluster de banco de dados `neptune_lab_mode`. Um índice OSGP só pode ser habilitado em um novo cluster de banco de dados vazio.

Por padrão, o índice OSGP está desabilitado.

> **Note**
>
> Depois de definir o parâmetro de cluster de banco de dados `neptune_lab_mode`para habilitar o índice OSGP, é necessário reiniciar a instância de gravador do cluster para que a alteração tenha efeito.

> **Warning**
>
> Se você desabilitar um índice OSGP habilitado configurando `ObjectIndex=disabled` e depois reabilitá-lo depois de adicionar mais dados, o índice não será criado corretamente. A reconstrução sob demanda do índice não é compatível, portanto, você só deve habilitar o índice OSGP quando o banco de dados estiver vazio.

Semântica de transação formalizada

O Neptune atualizou a semântica formal para transações simultâneas (consulte Semântica de transação no Neptune).

Use `ReadWriteConflictDetection` como o nome no parâmetro `neptune_lab_mode` que habilita ou desabilita a semântica de transação formalizada.

Por padrão, a semântica de transação formalizada já está habilitada. Se você desejar reverter para o comportamento anterior, inclua `ReadWriteConflictDetection=disabled` no valor definido para o parâmetro `neptune_lab_mode` do cluster de banco de dados.

Suporte estendido de data e hora

O Neptune estendeu o suporte para a funcionalidade de data e hora. Para habilitar a data e hora com formatos estendidos, `DatetimeMillisecond=enabled` inclua o `neptune_lab_mode` parâmetro DB Cluster no conjunto de valores.

O mecanismo de consulta alternativo (DFE) do Amazon Neptune

O Amazon Neptune tem um mecanismo de consulta alternativo conhecido como DFE, que usa recursos de instância de banco de dados, como núcleos de CPU, memória e E/S, com maior eficiência do que o mecanismo original do Neptune.

> **Note**
>
> Com grandes conjuntos de dados, o mecanismo DFE pode não funcionar bem em instâncias t3.

O mecanismo DFE executa consultas SPARQL, Gremlin e openCypher e é compatível com uma ampla variedade de tipos de plano, incluindo de profundidade esquerda, espessos e híbridos. Os operadores do plano podem invocar tanto as operações de computação, que são executadas em um conjunto reservado de núcleos de computação, quanto as operações de E/S, cada uma executada em seu próprio thread em um grupo de threads de E/S.

O DFE usa estatísticas pré-geradas sobre os dados do grafo do Neptune para tomar decisões embasadas sobre como estruturar consultas. Consulte Estatísticas do DFE para obter informações sobre como essas estatísticas são geradas.

A escolha do tipo de plano e do número de threads de computação usados é feita automaticamente com base em estatísticas pré-geradas e nos recursos que estão disponíveis no nó principal do Neptune. A ordem dos resultados não é predeterminada para planos com paralelismo computacional interno.

Controlar onde o mecanismo DFE do Neptune é usado

Por padrão, o parâmetro neptune_dfe_query_engine de uma instância é definido como `viaQueryHint`, o que faz com que o mecanismo DFE seja usado somente para consultas openCypher e para consultas Gremlin e SPARQL que incluam explicitamente a dica de consulta `useDFE` definida como `true`.

É possível habilitar totalmente o mecanismo DFE para que ele seja usado sempre que possível definindo o parâmetro de instância `neptune_dfe_query_engine` como `enabled`.

Você também pode desabilitar o DFE incluindo a dica de consulta `useDFE` para uma consulta do Gremlin ou do SPARQL específica. Essa dica de consulta permite impedir que o DFE execute essa consulta específica.

É possível determinar se o DFE está habilitado ou não em uma instância usando uma chamada Status de instância desta forma:

```
curl -G https://your-neptune-endpoint:port/status
```

Depois, a resposta de status especifica se o DFE está habilitado ou não:

```
{
  "status":"healthy",
  "startTime":"Wed Dec 29 02:29:24 UTC 2021",
  "dbEngineVersion":"development",
  "role":"writer",
  "dfeQueryEngine":"viaQueryHint",
  "gremlin":{"version":"tinkerpop-3.5.2"},
  "sparql":{"version":"sparql-1.1"},
  "opencypher":{"version":"Neptune-9.0.20190305-1.0"},
  "labMode":{
    "ObjectIndex":"disabled",
    "ReadWriteConflictDetection":"enabled"
  },
  "features":{
    "ResultCache":{"status":"disabled"},
    "IAMAuthentication":"disabled",
    "Streams":"disabled",
    "AuditLog":"disabled"
  },
  "settings":{"clusterQueryTimeoutInMs":"120000"}
}
```

Os resultados `explain` e `profile` do Gremlin informam se uma consulta está sendo executada pelo DFE. Consulte Informações contidas em um relatório `explain` do Gremlin no caso de `explain` e Relatórios de `profile` do DFE no caso de `profile`.

Da mesma forma, o `explain` do SPARQL informa se uma consulta SPARQL está sendo executada pelo DFE. Para obter mais detalhes, consulte Exemplo de saída de `explain` SPARQL quando o DFE está habilitado e Operador `DFENode`.

Construções de consulta compatíveis com o DFE do Neptune

No momento, o DFE do Neptune é compatível com um subconjunto de construções de consulta SPARQL e Gremlin.

No caso do SPARQL, trata-se do subconjunto de padrões de grafos básicos conjuntivos.

No caso do Gremlin, geralmente é o subconjunto de consultas que contêm uma cadeia de percursos que não apresentam algumas das etapas mais complexas.

Você pode descobrir se uma de suas consultas está sendo executada total ou parcialmente pelo DFE da seguinte maneira:

- No Gremlin, os resultados `explain` e `profile` informam quais partes de uma consulta estão sendo executadas pelo DFE, se houver. Consulte Informações contidas em um relatório `explain` do Gremlin no caso de `explain` e Relatórios de `profile` do DFE no caso de `profile`. Consulte também Ajustar consultas do Gremlin usando `explain` e `profile`.

  Detalhes sobre o suporte ao mecanismo do Neptune para etapas individuais do Gremlin estão documentados em Suporte a etapas do Gremlin.
- Da mesma forma, o `explain` do SPARQL informa se uma consulta SPARQL está sendo executada pelo DFE. Para obter mais detalhes, consulte Exemplo de saída de `explain` SPARQL quando o DFE está habilitado e Operador `DFENode`.

Gerenciar estatísticas a serem utilizadas pelo DFE do Neptune

> **Note**
>
> O suporte ao openCypher depende do mecanismo de consulta DFE no Neptune.
> O mecanismo DFE foi disponibilizado pela primeira vez no modo de laboratório na versão 1.0.3.0 do mecanismo do Neptune e, a partir da versão 1.0.5.0 do mecanismo do Neptune, ele se tornou habilitado por padrão, mas somente para uso com dicas de consulta e para suporte ao openCypher.
> A partir da versão 1.1.1.0 do mecanismo do Neptune, o mecanismo DFE não está mais no modo de laboratório e agora é controlado usando o parâmetro de instância neptune_dfe_query_engine no grupo de parâmetros de banco de dados de uma instância.

O mecanismo DFE usa informações sobre os dados no grafo do Neptune para fazer compensações efetivas ao planejar a execução da consulta. Essas informações assumem a forma de estatísticas que incluem os chamados conjuntos de características e estatísticas de predicados que podem orientar o planejamento de consultas.

A partir da versão 1.2.1.0 do mecanismo, você pode recuperar informações resumidas sobre seu gráfico a partir dessas estatísticas usando a API GetGraphSummary ou o endpoint. `summary`

No momento, essas estatísticas do DFE são geradas novamente sempre que mais de 10% dos dados no grafo são alterados ou quando as estatísticas mais recentes têm mais de dez dias. No entanto, esses gatilhos podem mudar no futuro.

> **Note**
>
> A geração de estatísticas está desabilitada em instâncias T3 e T4g porque pode exceder a capacidade de memória desses tipos de instância.

É possível gerenciar a geração de estatísticas do DFE por meio de um dos seguintes endpoints:

- `https://`*`your-neptune-host`*`:`*`port`*`/rdf/statistics` (para o SPARQL).
- `https://`*`your-neptune-host`*`:`*`port`*`/propertygraph/statistics` (para Gremlin e openCypher) e sua versão alternativa: `https://`*`your-neptune-host`*`:`*`port`*`/pg/statistics`.

> **Note**
>
> A partir da versão 1.1.1.0 do mecanismo, o endpoint de estatísticas do Gremlin (`https://your-neptune-host:port/gremlin/statistics`) está sendo descontinuado em favor do endpoint `propertygraph` ou `pg`. Ele ainda é compatível com versões anteriores, mas pode ser removido em versões futuras.
> A partir da versão 1.2.1.0 do mecanismo, o endpoint de estatísticas do SPARQL (`https://your-neptune-host:port/sparql/statistics`) está sendo descontinuado em favor do endpoint `rdf`. Ele ainda é compatível com versões anteriores, mas pode ser removido em versões futuras.

Nos exemplos abaixo, `$STATISTICS_ENDPOINT` significa qualquer um destes URLs de endpoint.

> **Note**
>
> Se um endpoint de estatísticas do DFE estiver em uma instância de leitor, as únicas solicitações que ele poderá processar serão solicitações de status. Outras solicitações falharão com uma `ReadOnlyViolationException`.

Limites de tamanho para geração de estatísticas do DFE

No momento, a geração de estatísticas do DFE será interrompida se um dos seguintes limites de tamanho for atingido:

- O número de conjuntos de características gerados não pode exceder cinquenta mil.
- O número de estatísticas de predicados geradas não pode exceder um milhão.

Esses limites podem mudar.

Status atual das estatísticas do DFE

É possível conferir o status atual das estatísticas do DFE usando a seguinte solicitação `curl`:

```
curl -G "$STATISTICS_ENDPOINT"
```

A resposta a uma solicitação de status contém os seguintes campos:

- `status`: o código de retorno HTTP da solicitação. Se a solicitação for bem-sucedida, o código será `200`. Para obter uma lista de erros comuns, consulte Erros comuns.
- `payload`:
  - `autoCompute`: (booliano) indica se a geração automática de estatísticas está habilitada ou não.
  - `active`: (booliano) indica se a geração automática de estatísticas do DFE está ou não habilitada.
  - `statisticsId` : relata o ID da execução atual da geração de estatísticas. Um valor de `-1` indica que nenhuma estatística foi gerada.
  - `date`: a hora UTC na qual as estatísticas do DFE foram geradas mais recentemente, no formato ISO 8601.

    > **Note**
    >
    > Antes da versão 1.2.1.0 do mecanismo, isso era representado com precisão de minutos, mas a partir da versão 1.2.1.0 do mecanismo, é representado com precisão de milissegundos (por exemplo, `2023-01-24T00:47:43.319Z`).

  - `note`: uma observação sobre problemas no caso em que as estatísticas são inválidas.
  - `signatureInfo`: contém informações sobre os conjuntos de características gerados nas estatísticas (antes da versão 1.2.1.0 do mecanismo, esse campo era denominado `summary`). Geralmente, eles não são diretamente úteis:
    - `signatureCount`: o número total de assinaturas em todos os conjuntos de características.
    - `instanceCount`: o número total de instâncias do conjunto de características.
    - `predicateCount`: o número total de predicados exclusivos.

A resposta a uma solicitação de status quando nenhuma estatística foi gerada é semelhante a esta:

```
{
  "status" : "200 OK",
  "payload" : {
    "autoCompute" : true,
    "active" : false,
    "statisticsId" : -1
   }
}
```

Se as estatísticas do DFE estiverem disponíveis, a resposta será semelhante a esta:

```
{
  "status" : "200 OK",
  "payload" : {
    "autoCompute" : true,
    "active" : true,
    "statisticsId" : 1588893232718,
    "date" : "2020-05-07T23:13Z",
    "summary" : {
      "signatureCount" : 5,
      "instanceCount" : 1000,
      "predicateCount" : 20
    }
  }
}
```

Se ocorrer uma falha na geração de estatísticas do DFE, por exemplo, porque excedeu o limite de tamanho das estatísticas, a resposta será semelhante a esta:

```
{
  "status" : "200 OK",
  "payload" : {
    "autoCompute" : true,
    "active" : false,
    "statisticsId" : 1588713528304,
    "date" : "2020-05-05T21:18Z",
    "note" : "Limit reached: Statistics are not available"
  }
}
```

Desabilitar a geração automática de estatísticas do DFE

Por padrão, a geração automática de estatísticas do DFE é habilitada quando você habilita o DFE.

É possível desabilitar a geração automática da seguinte forma:

```
curl -X POST "$STATISTICS_ENDPOINT" -d '{ "mode" : "disableAutoCompute" }'
```

Se a solicitação for bem-sucedida, o código de resposta HTTP será `200` e a resposta será:

```
{
```

```
  "status" : "200 OK"
}
```

É possível confirmar se a geração automática está desabilitada permitindo uma solicitação de status e conferindo se o campo `autoCompute` na resposta está definido como `false`.

A desativação da geração automática de estatísticas não encerra um cálculo estatístico em andamento.

Se você fizer uma solicitação para desabilitar a geração automática para uma instância de leitor e não para a instância de gravador do cluster de banco de dados, ocorrerá uma falha na solicitação com um código de retorno HTTP de 400 e uma saída como a seguinte:

```
{
  "detailedMessage" : "Writes are not permitted on a read replica instance",
  "code" : "ReadOnlyViolationException",
  "requestId":"8eb8d3e5-0996-4a1b-616a-74e0ec32d5f7"
}
```

Consulte Erros comuns para obter uma lista de outros erros comuns.

Reabilitar a geração automática de estatísticas do DFE

Por padrão, a geração automática de estatísticas do DFE já é habilitada quando você habilita o DFE. Se você desabilitar a geração automática, poderá reabilitá-la posteriormente da seguinte maneira:

```
curl -X POST "$STATISTICS_ENDPOINT" -d '{ "mode" : "enableAutoCompute" }'
```

Se a solicitação for bem-sucedida, o código de resposta HTTP será `200` e a resposta será:

```
{
  "status" : "200 OK"
}
```

É possível confirmar se a geração automática está habilitada permitindo uma solicitação de status e conferindo se o campo `autoCompute` na resposta está definido como `true`.

Acionar manualmente a geração de estatísticas do DFE

É possível iniciar a geração de estatísticas do DFE manualmente da seguinte forma:

```
curl -X POST "$STATISTICS_ENDPOINT" -d '{ "mode" : "refresh" }'
```

Se a solicitação for bem-sucedida, a saída será a seguinte, com um código de retorno HTTP de 200:

```
{
  "status" : "200 OK",
  "payload" : {
    "statisticsId" : 1588893232718
  }
}
```

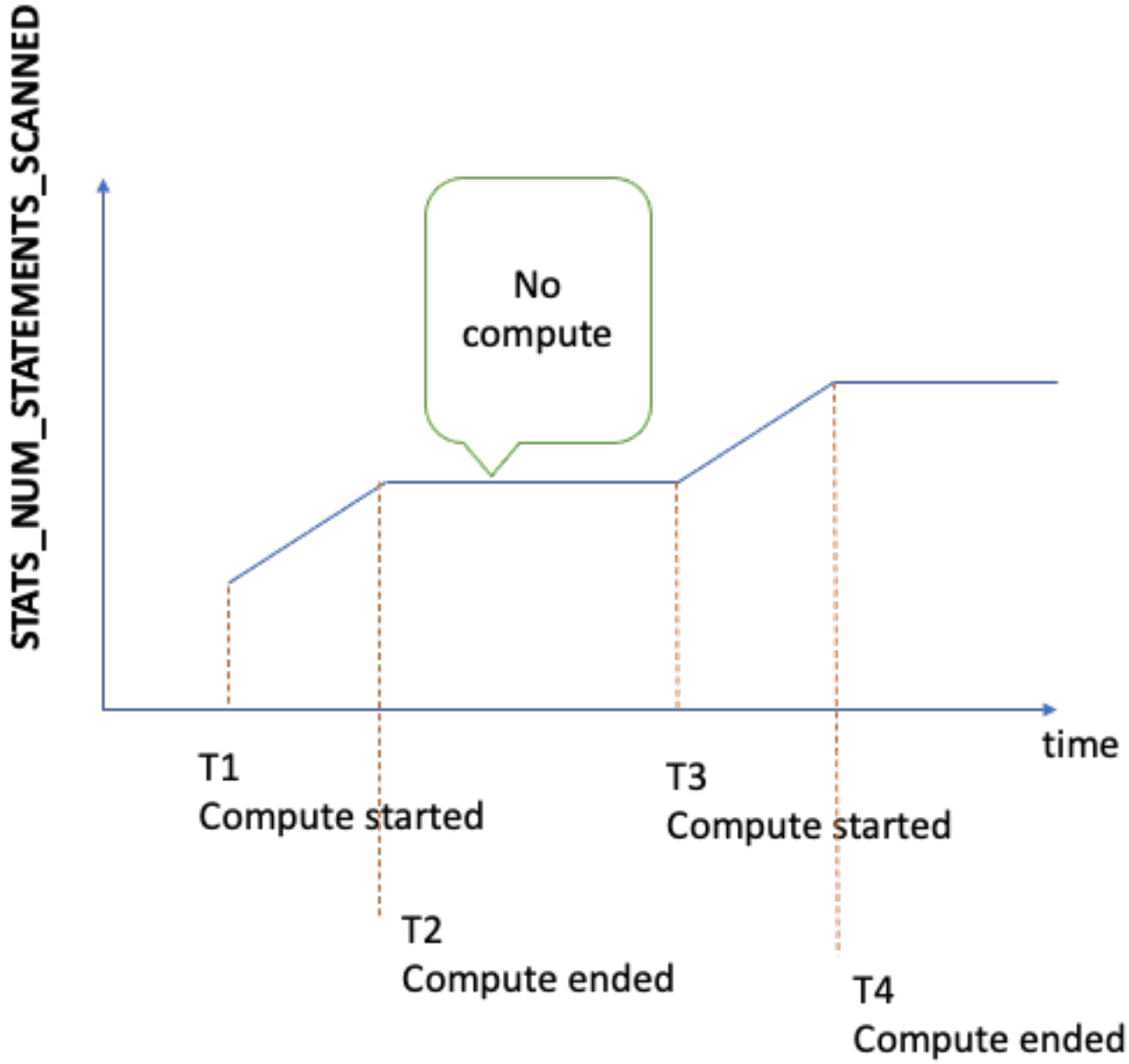
O `statisticsId` na saída é o ID da execução da geração de estatísticas que está ocorrendo atualmente. Se uma execução já estiver em andamento no momento da solicitação, a solicitação vai gerar o ID dessa execução em vez de iniciar outra. Somente uma execução de geração de estatísticas pode ocorrer por vez.

Se ocorrer um failover enquanto as estatísticas do DFE estiverem sendo geradas, o novo nó de gravador selecionará o último ponto de verificação processado e retomará a execução das estatísticas a partir daí.

Usando a **`StatsNumStatementsScanned`** CloudWatch métrica para monitorar o cálculo estatístico

A `StatsNumStatementsScanned` CloudWatch métrica retorna o número total de declarações escaneadas para cálculo estatístico desde a inicialização do servidor. Ela é atualizada em cada fatia de cálculo de estatísticas.

Toda vez que o cálculo de estatísticas é acionado, esse número aumenta e, quando nenhum cálculo está acontecendo, ele permanece constante. Portanto, a análise de um gráfico de valores `StatsNumStatementsScanned` ao longo do tempo oferece uma imagem bem clara de quando o cálculo de estatísticas estava acontecendo e com que rapidez:



Quando o cálculo está ocorrendo, a inclinação do grafo mostra a velocidade (quanto mais íngreme a inclinação, com maior rapidez as estatísticas são calculadas).

Se o grafo for simplesmente uma linha plana em 0, o atributo de estatísticas foi habilitado, mas nenhuma estatística foi calculada. Se o atributo de estatísticas tiver sido desativado ou se você estiver usando uma versão do mecanismo que não é compatível com o cálculo de estatísticas, `StatsNumStatementsScanned` não existe.

Conforme mencionado anteriormente, é possível desabilitar o cálculo de estatísticas usando a API de estatísticas, mas deixá-lo desabilitado pode fazer com que as estatísticas não estejam atualizadas,

o que, por sua vez, pode ocasionar a geração deficiente do plano de consulta para o mecanismo do DFE.

Consulte Monitorando Neptune usando a Amazon CloudWatch para obter informações sobre como usar CloudWatch.

Usando a autenticação AWS Identity and Access Management (IAM) com endpoints de estatísticas do DFE

É possível acessar os endpoints de estatísticas do DFE de forma segura com a autenticação do IAM usando awscurl ou qualquer outra ferramenta que funcione com HTTPS e IAM. Consulte Usar `awscurl` com credenciais temporárias para se conectar com segurança a um cluster de banco de dados com a autenticação do IAM habilitada para saber como configurar as credenciais adequadas. Depois de fazer isso, é possível fazer uma solicitação de status como esta:

```
awscurl "$STATISTICS_ENDPOINT" \
    --region (your region) \
    --service neptune-db
```

Ou, por exemplo, você pode criar um arquivo JSON denominado `request.json` que contém:

```
{ "mode" : "refresh" }
```

Depois, é possível iniciar a geração de estatísticas manualmente da seguinte forma:

```
awscurl "$STATISTICS_ENDPOINT" \
    --region (your region) \
    --service neptune-db \
    -X POST -d @request.json
```

Excluir estatísticas do DFE

É possível excluir todas as estatísticas no banco de dados fazendo uma solicitação HTTP DELETE para o endpoint de estatísticas:

```
curl -X "DELETE" "$STATISTICS_ENDPOINT"
```

Os códigos de retorno HTTP válidos são:

- `200`: a exclusão foi bem-sucedida.

Nesse caso, uma resposta típica seria:

```
{
  "status" : "200 OK",
  "payload" : {
      "active" : false,
      "statisticsId" : -1
  }
}
```

- `204`: não havia estatísticas para excluir.

  Nesse caso, a resposta está em branco (sem resposta).

Se você enviar uma solicitação de exclusão a um endpoint de estatísticas em um nó de leitor, uma `ReadOnlyViolationException` será lançada.

Códigos de erro comuns para solicitação de estatísticas do DFE

Veja a seguir uma lista de erros comuns que podem ocorrer quando você faz uma solicitação a um endpoint de estatísticas:

- `AccessDeniedException`: Código de retorno: `400`. Mensagem: `Missing Authentication Token`.
- `BadRequestException` (para Gremlin e openCypher): Código de retorno: `400`. Mensagem: `Bad route: /pg/statistics`.
- `BadRequestException` (para dados do RDF): Código de retorno: `400`. Mensagem: `Bad route: /rdf/statistics`.
- `InvalidParameterException`: Código de retorno: `400`. Mensagem: `Statistics command parameter 'mode' has unsupported value '`*`the invalid value`*`'`.
- `MissingParameterException`: Código de retorno: `400`. Mensagem: `Content-type header not specified.`.
- `ReadOnlyViolationException`: Código de retorno: `400`. Mensagem: `Writes are not permitted on a read replica instance`.

Por exemplo, se você fizer uma solicitação quando o DFE e as estatísticas não estiverem habilitados, receberá uma resposta como a seguinte:

```
{
  "code" : "BadRequestException",
  "requestId" : "b2b8f8ee-18f1-e164-49ea-836381a3e174",
  "detailedMessage" : "Bad route: /sparql/statistics"
}
```

Obter um relatório resumido rápido sobre o grafo

A API de resumo do grafo do Neptune recupera as seguintes informações sobre o grafo:

- Para grafos de propriedades (PG), a API de resumo do grafo gera uma lista somente leitura de rótulos de nós e bordas e chaves de propriedade, junto com contagens de nós, bordas e propriedades.
- Para grafos do framework de descrição de recursos (RDF), a API de resumo do grafo gera uma lista somente leitura de classes e chaves de predicados, junto com contagens de quádruplos, assuntos e predicados.

> **Note**
>
> A API de resumo do grafo foi introduzida na versão 1.2.1.0 do mecanismo do Neptune.

Com a API de resumo do grafo, é possível obter rapidamente uma compreensão geral do tamanho e do conteúdo dos dados do grafo. Você também pode usar a API de forma interativa em um caderno do Neptune usando a magia `%summary` da bancada de trabalho do Neptune. Em uma aplicação de grafos, a API pode ser usada para melhorar os resultados da pesquisa fornecendo rótulos de nós ou bordas descobertos como parte da pesquisa.

Os dados de resumo do grafo são extraídos das estatísticas do DFE calculadas pelo mecanismo DFE do Neptune durante o runtime e estarão disponíveis sempre que as estatísticas do DFE estiverem disponíveis. Por padrão, as estatísticas são habilitadas quando você cria um cluster de banco de dados do Neptune.

> **Note**
>
> A geração de estatísticas está desabilitada nos tipos de instância `t3` e `t4` (ou seja, nos tipos `db.t3.medium` e `db.t4g.medium`) para conservar a memória. Como resultado, os dados de resumo do gráfico também não estão disponíveis nesses tipos de instância.

Você pode conferir o status das estatísticas do DFE usando a API de status de estatísticas. Desde que a geração automática de estatísticas não tenha sido desabilitada, as estatísticas serão atualizadas automaticamente de forma periódica.

Se você quiser ter certeza de que as estatísticas estão o mais atualizadas possível ao solicitar um resumo do grafo, acione manualmente uma atualização de estatísticas logo antes de recuperar o resumo. Se o grafo estiver mudando enquanto as estatísticas estiverem sendo calculadas, elas ficarão um pouco atrasadas, mas não muito.

Usar a API de resumo do grafo para recuperar informações resumidas do grafo

Para um grafo de propriedades que você consulta usando o Gremlin ou o openCypher, é possível recuperar um resumo do grafo a partir do endpoint de resumo do grafo de propriedades. Há um URI longo e um curto para esse endpoint:

- `https://`*`your-neptune-host`*`:`*`port`*`/propertygraph/statistics/summary`
- `https://`*`your-neptune-host`*`:`*`port`*`/pg/statistics/summary`

Para um grafo do RDF consultado com o SPARQL, é possível recuperar um resumo do grafo do endpoint de resumo do RDF:

- `https://`*`your-neptune-host`*`:`*`port`*`/rdf/statistics/summary`

Esses endpoints são somente leitura e são compatíveis somente com uma operação `GET` HTTP. Se $GRAPH_SUMMARY_ENDPOINT estiver definido como o endereço de qualquer endpoint que você queira consultar, poderá recuperar os dados do resumo usando `curl` e `GET` HTTP da seguinte forma:

```
curl -G "$GRAPH_SUMMARY_ENDPOINT"
```

Se nenhuma estatística estiver disponível quando você tentar recuperar um resumo do grafo, a resposta será semelhante a esta:

```
{
  "detailedMessage": "Statistics are not available. Summary can only be generated after
 statistics are available.",
  "requestId": "48c1f788-f80b-b69c-d728-3f6df579a5f6",
  "code": "StatisticsNotAvailableException"
}
```

O parâmetro de consulta de URL **`mode`** para a API de resumo do grafo

A API de resumo do grafo aceita um parâmetro de consulta de URL chamado `mode`, que pode ter um dos dois valores, a saber, `basic` (o padrão) e `detailed`. Para um grafo do RDF, a resposta do resumo do grafo do modo `detailed` contém um campo `subjectStructures` adicional. Para um grafo de propriedades, a resposta detalhada do resumo do grafo contém dois campos adicionais, a saber, `nodeStructures` e `edgeStructures`.

Para solicitar uma resposta do resumo do grafo `detailed`, inclua o parâmetro `mode` da seguinte forma:

```
curl -G "$GRAPH_SUMMARY_ENDPOINT?mode=detailed"
```

Se o parâmetro `mode` não estiver presente, o modo `basic` será usado por padrão, portanto, embora seja possível especificar `?mode=basic` explicitamente, isso não é necessário.

Resposta do resumo de um grafo de propriedades (PG)

Para um grafo de propriedades em branco, a resposta detalhada do resumo do grafo é semelhante à seguinte:

```
{
  "status" : "200 OK",
  "payload" : {
    "version" : "v1",
    "lastStatisticsComputationTime" : "2023-01-10T07:58:47.972Z",
    "graphSummary" : {
      "numNodes" : 0,
      "numEdges" : 0,
      "numNodeLabels" : 0,
      "numEdgeLabels" : 0,
      "nodeLabels" : [ ],
      "edgeLabels" : [ ],
      "numNodeProperties" : 0,
      "numEdgeProperties" : 0,
      "nodeProperties" : [ ],
      "edgeProperties" : [ ],
      "totalNodePropertyValues" : 0,
      "totalEdgePropertyValues" : 0,
      "nodeStructures" : [ ],
      "edgeStructures" : [ ]
```

```
    }
  }
}
```

Uma resposta do resumo do grafo de propriedades (PG) tem os seguintes campos:

- **`status`**: o código de retorno HTTP da solicitação. Se a solicitação for bem-sucedida, o código será 200.

  Para obter uma lista de erros comuns, consulte Erros comuns de resumo do grafo.
- **`payload`**
  - **`version`**: a versão dessa resposta do resumo do grafo.
  - **`lastStatisticsComputationTime`** : a data e hora, no formato ISO 8601, da hora em que o Neptune calculou as estatísticas pela última vez.
  - **`graphSummary`**
    - **`numNodes`**: o número de nós no grafo.
    - **`numEdges`**: o número de bordas no grafo.
    - **`numNodeLabels`**: o número de rótulos de nós distintos no grafo.
    - **`numEdgeLabels`**: o número de rótulos de bordas distintos no grafo.
    - **`nodeLabels`**: lista de rótulos de nós distintos no grafo.
    - **`edgeLabels`**: lista de rótulos de bordas distintos no grafo.
    - **`numNodeProperties`**: o número de propriedades de nós distintas no grafo.
    - **`numEdgeProperties`**: o número de propriedades de bordas distintas no grafo.
    - **`nodeProperties`**: lista de propriedades de nós distintas no grafo, junto com a contagem de nós em que cada propriedade é usada.
    - **`edgeProperties`**: lista de propriedades de bordas distintas no grafo, junto com a contagem de bordas em que cada propriedade é usada.
    - **`totalNodePropertyValues`**: número total de usos de todas as propriedades de nós.
    - **`totalEdgePropertyValues`**: número total de usos de todas as propriedades de bordas.
    - **`nodeStructures`**: esse campo só está presente quando *`mode=detailed`* está especificado na solicitação. Ele contém uma lista de estruturas de nós, cada uma contendo os seguintes campos:
      - **`count`**: número de nós que têm essa estrutura específica.
      - **`nodeProperties`**: lista das propriedades dos nós presentes nessa estrutura específica.

    - **`distinctOutgoingEdgeLabels`**: lista de rótulos de borda de saída distintos presentes nessa estrutura específica.
  - **`edgeStructures`**: esse campo só está presente quando *`mode=detailed`* está especificado na solicitação. Ele contém uma lista de estruturas de bordas, cada uma contendo os seguintes campos:
    - **`count`**: número de bordas que têm essa estrutura específica.
    - **`edgeProperties`**: lista das propriedades das bordas presentes nessa estrutura específica.

Resposta do resumo de um grafo do RDF

Para um grafo do RDF em branco, a resposta detalhada do resumo do grafo é semelhante à seguinte:

```
{
  "status" : "200 OK",
  "payload" : {
    "version" : "v1",
    "lastStatisticsComputationTime" : "2023-01-10T07:58:47.972Z",
    "graphSummary" : {
      "numDistinctSubjects" : 0,
      "numDistinctPredicates" : 0,
      "numQuads" : 0,
      "numClasses" : 0,
      "classes" : [ ],
      "predicates" : [ ],
      "subjectStructures" : [ ]
    }
  }
}
```

Uma resposta do resumo do grafo do RDF tem os seguintes campos:

- **`status`**: o código de retorno HTTP da solicitação. Se a solicitação for bem-sucedida, o código será 200.

  Para obter uma lista de erros comuns, consulte Erros comuns de resumo do grafo.
- **`payload`**
  - **`version`**: a versão dessa resposta do resumo do grafo.

- **`lastStatisticsComputationTime`** : a data e hora, no formato ISO 8601, da hora em que o Neptune calculou as estatísticas pela última vez.
- **`graphSummary`**
  - **`numDistinctSubjects`**: o número de assuntos distintos no grafo.
  - **`numDistinctPredicates`**: o número de predicados distintos no grafo.
  - **`numQuads`**: o número de quadrantes no grafo.
  - **`numClasses`**: o número de classes no grafo.
  - **`classes`**: lista de classes no grafo.
  - **`predicates`**: lista de predicados no grafo, junto com as contagens de predicados.
  - **`subjectStructures`**: esse campo só está presente quando *`mode=detailed`* está especificado na solicitação. Ele contém uma lista de estruturas de assuntos, cada uma contendo os seguintes campos:
    - **`count`**: número de ocorrências dessa estrutura específica.
    - **`predicates`**: lista das predicados presentes nessa estrutura específica.

Exemplo de resposta do resumo do grafo de propriedades (PG)

Aqui está a resposta do resumo detalhada para um grafo de propriedades que contém o exemplo do conjunto de dados de rotas aéreas do grafo de propriedades:

```
{
  "status" : "200 OK",
  "payload" : {
    "version" : "v1",
    "lastStatisticsComputationTime" : "2023-03-01T14:35:03.804Z",
    "graphSummary" : {
      "numNodes" : 3748,
      "numEdges" : 51300,
      "numNodeLabels" : 4,
      "numEdgeLabels" : 2,
      "nodeLabels" : [
        "continent",
        "country",
        "version",
        "airport"
      ],
      "edgeLabels" : [
```

```
        "contains",
        "route"
      ],
      "numNodeProperties" : 14,
      "numEdgeProperties" : 1,
      "nodeProperties" : [
        {
          "desc" : 3748
        },
        {
          "code" : 3748
        },
        {
          "type" : 3748
        },
        {
          "country" : 3503
        },
        {
          "longest" : 3503
        },
        {
          "city" : 3503
        },
        {
          "lon" : 3503
        },
        {
          "elev" : 3503
        },
        {
          "icao" : 3503
        },
        {
          "region" : 3503
        },
        {
          "runways" : 3503
        },
        {
          "lat" : 3503
        },
        {
          "date" : 1
```

```
        },
        {
          "author" : 1
        }
      ],
      "edgeProperties" : [
        {
          "dist" : 50532
        }
      ],
      "totalNodePropertyValues" : 42773,
      "totalEdgePropertyValues" : 50532,
      "nodeStructures" : [
        {
          "count" : 3471,
          "nodeProperties" : [
            "city",
            "code",
            "country",
            "desc",
            "elev",
            "icao",
            "lat",
            "lon",
            "longest",
            "region",
            "runways",
            "type"
          ],
          "distinctOutgoingEdgeLabels" : [
            "route"
          ]
        },
        {
          "count" : 161,
          "nodeProperties" : [
            "code",
            "desc",
            "type"
          ],
          "distinctOutgoingEdgeLabels" : [
            "contains"
          ]
        },
```

```
        {
          "count" : 83,
          "nodeProperties" : [
            "code",
            "desc",
            "type"
          ],
          "distinctOutgoingEdgeLabels" : [ ]
        },
        {
          "count" : 32,
          "nodeProperties" : [
            "city",
            "code",
            "country",
            "desc",
            "elev",
            "icao",
            "lat",
            "lon",
            "longest",
            "region",
            "runways",
            "type"
          ],
          "distinctOutgoingEdgeLabels" : [ ]
        },
        {
          "count" : 1,
          "nodeProperties" : [
            "author",
            "code",
            "date",
            "desc",
            "type"
          ],
          "distinctOutgoingEdgeLabels" : [ ]
        }
      ],
      "edgeStructures" : [
        {
          "count" : 50532,
          "edgeProperties" : [
            "dist"
```

```
          ]
        }
      ]
    }
  }
}
```

Exemplo de resposta do resumo do grafo do RDF

Veja a resposta do resumo detalhada para um grafo do RDF que contém o exemplo do conjunto de dados de rotas aéreas do RDF:

```
{
  "status" : "200 OK",
  "payload" : {
    "version" : "v1",
    "lastStatisticsComputationTime" : "2023-03-01T14:54:13.903Z",
    "graphSummary" : {
      "numDistinctSubjects" : 54403,
      "numDistinctPredicates" : 19,
      "numQuads" : 158571,
      "numClasses" : 4,
      "classes" : [
        "http://kelvinlawrence.net/air-routes/class/Version",
        "http://kelvinlawrence.net/air-routes/class/Airport",
        "http://kelvinlawrence.net/air-routes/class/Continent",
        "http://kelvinlawrence.net/air-routes/class/Country"
      ],
      "predicates" : [
        {
          "http://kelvinlawrence.net/air-routes/objectProperty/route" : 50656
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/dist" : 50656
        },
        {
          "http://kelvinlawrence.net/air-routes/objectProperty/contains" : 7004
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/code" : 3747
        },
        {
          "http://www.w3.org/2000/01/rdf-schema#label" : 3747
```

```
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/type" : 3747
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/desc" : 3747
        },
        {
          "http://www.w3.org/1999/02/22-rdf-syntax-ns#type" : 3747
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/icao" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/lat" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/region" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/runways" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/longest" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/elev" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/lon" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/country" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/city" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/author" : 1
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/date" : 1
        }
      ],
```

```
    "subjectStructures" : [
      {
        "count" : 50656,
        "predicates" : [
          "http://kelvinlawrence.net/air-routes/datatypeProperty/dist"
        ]
      },
      {
        "count" : 3471,
        "predicates" : [
          "http://kelvinlawrence.net/air-routes/datatypeProperty/city",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/code",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/country",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/desc",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/elev",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/icao",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/lat",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/lon",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/longest",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/region",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/runways",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/type",
          "http://kelvinlawrence.net/air-routes/objectProperty/route",
          "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
          "http://www.w3.org/2000/01/rdf-schema#label"
        ]
      },
      {
        "count" : 238,
        "predicates" : [
          "http://kelvinlawrence.net/air-routes/datatypeProperty/code",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/desc",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/type",
          "http://kelvinlawrence.net/air-routes/objectProperty/contains",
          "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
          "http://www.w3.org/2000/01/rdf-schema#label"
        ]
      },
      {
        "count" : 31,
        "predicates" : [
          "http://kelvinlawrence.net/air-routes/datatypeProperty/city",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/code",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/country",
```

```
            "http://kelvinlawrence.net/air-routes/datatypeProperty/desc",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/elev",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/icao",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/lat",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/lon",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/longest",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/region",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/runways",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/type",
            "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
            "http://www.w3.org/2000/01/rdf-schema#label"
          ]
        },
        {
          "count" : 6,
          "predicates" : [
            "http://kelvinlawrence.net/air-routes/datatypeProperty/code",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/desc",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/type",
            "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
            "http://www.w3.org/2000/01/rdf-schema#label"
          ]
        },
        {
          "count" : 1,
          "predicates" : [
            "http://kelvinlawrence.net/air-routes/datatypeProperty/author",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/code",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/date",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/desc",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/type",
            "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
            "http://www.w3.org/2000/01/rdf-schema#label"
          ]
        }
      ]
    }
  }
}
```

Usando a autenticação AWS Identity and Access Management (IAM) com endpoints de resumo gráfico

É possível acessar os endpoints do resumo do grafo de forma segura com a autenticação do IAM usando awscurl ou qualquer outra ferramenta que funcione com HTTPS e IAM. Consulte Usar `awscurl` com credenciais temporárias para se conectar com segurança a um cluster de banco de dados com a autenticação do IAM habilitada para saber como configurar as credenciais adequadas. Depois de fazer isso, é possível fazer solicitações como esta:

```
awscurl "$GRAPH_SUMMARY_ENDPOINT" \
    --region (your region) \
    --service neptune-db
```

> **Important**
>
> A identidade ou função do IAM que cria as credenciais temporárias deve ter uma política do IAM anexada que permita a ação GetGraphsumária do IAM.

Consulte Erros de autenticação do IAM para obter uma lista de erros comuns do IAM que você pode encontrar.

Códigos de erro comuns que uma solicitação de resumo do grafo pode gerar

Código de erro do serviço do Neptune	Statu HTT	Message	Cenário de erro	Atenuação
`AccessDeniedException`	403	Token de autenticação ausente.	A solicitação não assinada ou assinada incorretamente foi enviada ao banco de dados Neptune com o IAM habilitado.	Assine a solicitação com o SigV4 antes de enviar (consulte IAM e resumos dos grafos).

Código de erro do serviço do Neptune	Statu HTT	Message	Cenário de erro	Atenuação
	403	*Usuário: (ARN do usuário) não está autorizado a executar: neptune-db: no recurso: (ARN do GetGraphSummary recurso).*	A política do IAM não permite a ação GetGraphSummary quando a solicitação de resumo do gráfico foi enviada ao banco de dados Neptune com o IAM ativado.	Garanta que a política do IAM vinculada ao usuário ou ao perfil que realiza a solicitação permite a ação `GetGraphSummary` .
`BadRequestException`	400	As estatísticas estão desabilitadas, portanto, o resumo do grafo também está desabilitado.	Tentar obter um resumo dos tipos de instância com capacidade de intermitência (`t3` ou `t4g`) em que as estatísticas estão desabilitadas.	Use um tipo de instância em que a geração de estatísticas esteja habilitada (todas as instâncias compatíveis, exceto `t3` e `t4g`).
	400	Rota inadequada: */rdf/statistics/summarypathapi*	Solicitação enviada a um caminho inválido.	Use a rota correta para o endpoint de resumo do grafo.
`InvalidParameterException`	400	A solicitação contém parâmetros desconhecidos: "*(parâmetro ou parâmetros desconhecidos)* ".	Quando um parâmetro inválido é especificado na solicitação.	Use somente parâmetros válidos (como `mode`) na solicitação.

Código de erro do serviço do Neptune	Statu HTT	Message	Cenário de erro	Atenuação
InvalidParameterException	400	O parâmetro de consulta de URI “mode” tem um valor não compatível “*(valor inválido)* ”.	Quando o parâmetro de URL “mode” na solicitação é seguido por um valor inválido.	Use valores válidos (como `basic` ou `detailed`) ao especificar o parâmetro de URL “mode”.
MethodNotAllowedException	405	Método não permitido.	Chamar o endpoint do resumo com qualquer método HTTP diferente de `GET` (como `POST` ou `DELETE`).	Use o método `GET` HTTP ao chamar o endpoint de resumo.
StatisticsNotAvailableException	400	As estatísticas ainda não foram calculadas, o resumo do grafo estará disponível após a conclusão do cálculo das estatísticas.	Não há estatísticas disponíveis quando a solicitação é enviada ao endpoint de resumo.	Espere até que a geração das estatísticas seja concluída. Você pode conferir o status da geração das estatísticas usando a API de status de estatísticas.
	400	Limite de estatísticas atingido, portanto, o resumo do grafo não está disponível.	A geração de estatísticas foi interrompida porque atingiu os limites de tamanho das estatísticas.	O resumo do grafo não está disponível nesse grafo.

Por exemplo, se você fizer uma solicitação para representar graficamente o endpoint de resumo em um banco de dados do Neptune com a autenticação do IAM habilitada e as permissões necessárias

não estiverem presentes na política do IAM do solicitante, você obterá uma resposta como a seguinte:

```
{
  "detailedMessage": "User: arn:aws:iam::(account ID):(user or user name) is not
 authorized to perform: neptune-db:GetGraphSummary on resource: arn:aws:neptune-
db:(region):(account ID):(cluster resource ID)/*",
  "requestId": "7ac2b98e-b626-d239-1d05-74b4c88fce82",
  "code": "AccessDeniedException"
}
```

Conectividade JDBC do Amazon Neptune

O Amazon Neptune lançou um driver JDBC de código aberto que é compatível com consultas do openCypher, do Gremlin, do SQL-Gremlin e do SPARQL. A conectividade JDBC facilita a conexão com o Neptune com ferramentas de business intelligence (BI), como o Tableau. Não há custo adicional em usar o driver JDBC com o Neptune. Você ainda paga somente pelos recursos consumidos do Neptune.

O driver é compatível com o JDBC 4.2 e requer pelo menos o Java 8. Para obter mais informações sobre o JDBC, consulte a documentação de API JDBC.

O GitHub projeto, no qual você pode registrar problemas e abrir solicitações de recursos, contém documentação detalhada do driver:

Driver JDBC para Amazon Neptune

- Usar SQL com o driver JDBC
- Usar o Gremlin com o driver JDBC
- Usar o openCypher com o driver JDBC
- Usar o SPARQL com o driver JDBC

Conceitos básicos do driver JDBC do Neptune

Para usar o driver JDBC do Neptune para se conectar a uma instância do Neptune, o driver JDBC deve ser implantado em uma instância do Amazon EC2 na mesma VPC do seu cluster de banco de dados do Neptune, ou a instância deve estar disponível por meio de um túnel SSH ou balanceador de carga. Um túnel SSH pode ser configurado internamente no driver ou externamente.

Você pode baixar o driver aqui. O driver vem embalado como um único arquivo JAR com um nome como `neptune-jdbc-1.0.0-all.jar`. Para usá-lo, coloque o arquivo JAR no `classpath` de sua aplicação. Ou, se a aplicação usa Maven ou Gradle, é possível usar os comandos apropriados do Maven ou do Gradle para instalar o driver pelo JAR.

O driver precisa de um URL de conexão JDBC para se conectar ao Neptune, em um formato como este:

```
jdbc:neptune:(connection
 type)://(host);property=value;property=value;...;property=value
```

As seções de cada linguagem de consulta no GitHub projeto descrevem as propriedades que você pode definir na URL de conexão do JDBC para essa linguagem de consulta.

Se o arquivo JAR estiver no `classpath` da aplicação, nenhuma outra configuração será necessária. Você pode conectar o driver usando a interface `DriverManager` JDBC e uma string de conexão do Neptune. Por exemplo, se o cluster de banco de dados do Neptune estiver acessível por meio do endpoint `neptune-example.com` na porta 8182, você poderá se conectar ao openCypher da seguinte forma:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

void example() {
    String url = "jdbc:neptune:opencypher://bolt://neptune-example:8182";

    Connection connection = DriverManager.getConnection(url);
    Statement statement = connection.createStatement();

    connection.close();
}
```

As seções de documentação do GitHub projeto para cada linguagem de consulta descrevem como construir a cadeia de conexão ao usar essa linguagem de consulta.

Usar o Tableau com o driver JDBC do Neptune

Para usar o Tableau com o driver JDBC do Neptune, comece baixando e instalando a versão mais recente do Tableau Desktop. Baixe o arquivo JAR para o driver JDBC do Neptune e também o arquivo do conector Tableau do Neptune (um arquivo `.taco`).

Como se conectar ao Tableau para Neptune em um Mac

1. Coloque o arquivo JAR do driver JDBC do Neptune na pasta `/Users/`*`(your user name)`*`/Library/Tableau/Drivers`.
2. Coloque o arquivo do conector `.taco` Tableau do Neptune na pasta `/Users/`*`(your user name)`*`/Documents/My Tableau Repository/Connectors`.
3. Se você tiver a autenticação do IAM habilitada, configure o ambiente para ela. Observe que as variáveis de ambiente definidas em `.zprofile/`, `.zshenv/`, `.bash_profile`, etc. não

funcionarão. As variáveis de ambiente devem ser definidas para que possam ser carregadas por uma aplicação de GUI.

Uma forma de definir suas credenciais é colocar sua chave de acesso e chave secreta no arquivo `/Users/(your user name)/.aws/credentials`.

Uma maneira fácil de definir a região de serviço é abrir um terminal e digitar o seguinte comando, usando a região da aplicação (por exemplo, `us-east-1`):

```
launchctl setenv SERVICE_REGION region name
```

Há outras maneiras de definir variáveis de ambiente que são persistidas após uma reinicialização, mas qualquer técnica usada deve definir variáveis que sejam acessíveis a uma aplicação de GUI.

4. Para fazer com que as variáveis de ambiente sejam carregadas em uma GUI no Mac, digite este comando em um terminal:

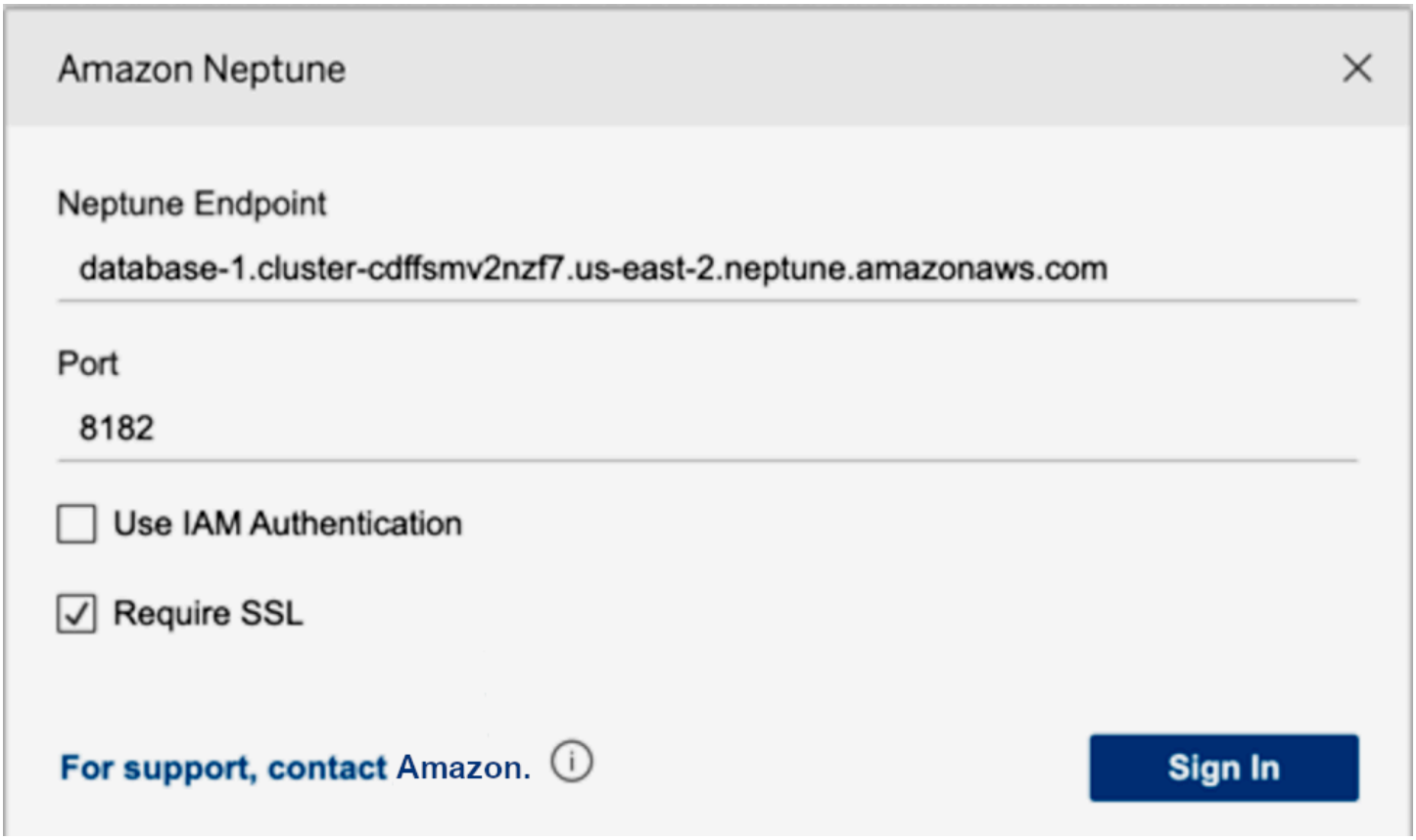
```
/Applications/Tableau/Desktop/2021.1.app/Contents/MacOS/Tableau
```

Como se conectar ao Tableau para Neptune em uma máquina Windows

1. Coloque o arquivo JAR do driver JDBC do Neptune na pasta `C:\Program Files\Tableau\Drivers`.
2. Coloque o arquivo do conector `.taco` Tableau do Neptune na pasta `C:\Users\(your user name)\Documents\My Tableau Repository\Connectors`.
3. Se você tiver a autenticação do IAM habilitada, configure o ambiente para ela.

   Isso pode ser tão simples quanto definir as variáveis de ambiente `ACCESS_KEY`, `SECRET_KEY` e `SERVICE_REGION` do usuário.

Com o Tableau aberto, selecione Mais no lado esquerdo da janela. Se o arquivo do conector do Tableau estiver localizado corretamente, você poderá selecionar Amazon Neptune da AWS na lista exibida:



Não é necessário editar a porta nem adicionar nenhuma opção de conexão. Insira o endpoint do Neptune e defina a configuração do SSL e do IAM (será necessário habilitar o SSL se estiver usando o IAM).

Ao selecionas Entrar, poderá levar mais de 30 segundos para se conectar se o grafo for grande. O Tableau está coletando tabelas de vértices e bordas e unindo vértices nas bordas, além de criar visualizações.

Solucionar problemas de conexão com o driver JDBC

Se o driver não conseguir se conectar ao servidor, use a função `isValid` do objeto `Connection` JDBC para conferir se a conexão é válida. Se a função gerar `false`, o que significa que a conexão é inválida, confira se o endpoint ao qual está conectado está correto e se você está na VPC do cluster de banco de dados do Neptune ou se tem um túnel SSH válido para o cluster.

Se você receber uma resposta `No suitable driver found for` *`(connection string)`* da chamada `DriverManager.getConnection`, é provável que haja um problema no início da string de conexão. Garanta que a string de conexão comece assim:

```
jdbc:neptune:opencypher://...
```

Para coletar mais informações sobre a conexão, você pode adicionar `LogLevel` à string de conexão da seguinte forma:

```
jdbc:neptune:opencypher://(JDBC URL):(port);logLevel=trace
```

Como alternativa, é possível adicionar `properties.put("logLevel", "trace")` às propriedades de entrada para registrar informações de rastreamento.

Atualizações do mecanismo do Amazon Neptune

O Amazon Neptune lança atualizações do mecanismo regularmente. Você pode determinar qual versão do mecanismo você instalou atualmente usando a API de status da instância.

As versões do mecanismo são listadas em Versões do mecanismo do Amazon Neptune, e os patches são listados em Atualizações mais recentes.

É possível encontrar mais informações sobre como as atualizações são lançadas e como atualizar o mecanismo do Neptune no banco de dados em Manutenção do cluster. Por exemplo, a numeração das versões é explicada em Números das versões do mecanismo.

Segurança no Amazon Neptune

A segurança na nuvem AWS é a maior prioridade. Como AWS cliente, você se beneficia de uma arquitetura de data center e rede criada para atender aos requisitos das organizações mais sensíveis à segurança.

A segurança é uma responsabilidade compartilhada entre você AWS e você. O modelo de responsabilidade compartilhada descreve isto como segurança da nuvem e segurança na nuvem:

- Segurança da nuvem — AWS é responsável por proteger a infraestrutura que executa AWS os serviços na AWS nuvem. AWS também fornece serviços que você pode usar com segurança. Auditores de terceiros testam e verificam regularmente a eficácia da nossa segurança como parte dos programas de conformidade da AWS. Para saber mais sobre os programas de conformidade que se aplicam ao Amazon Neptune, consulte Serviços da AWS no escopo por programa de conformidade.
- Segurança na nuvem — Sua responsabilidade é determinada pelo AWS serviço que você usa. Você também é responsável por outros fatores, incluindo a confidencialidade de seus dados, os requisitos da empresa e as leis e regulamentos aplicáveis.

Essa documentação ajuda a entender como aplicar o modelo de responsabilidade compartilhada ao usar o Neptune. Os tópicos a seguir mostram como configurar o Neptune para atender aos seus objetivos de segurança e conformidade. Você também aprenderá a usar outros AWS serviços que ajudam a monitorar e proteger seus recursos do Neptune.

Tópicos

- Resiliência no Amazon Neptune

Proteção de dados no Amazon Neptune

O modelo de responsabilidade AWS compartilhada se aplica à proteção de dados no Amazon Neptune. Conforme descrito neste modelo, AWS é responsável por proteger a infraestrutura global que executa todos os Nuvem AWS. Você é responsável por manter o controle sobre seu conteúdo hospedado nessa infraestrutura. Você também é responsável pelas tarefas de configuração e gerenciamento de segurança dos Serviços da AWS que usa. Para obter mais informações sobre a privacidade de dados, consulte as Perguntas Frequentes sobre Privacidade de Dados.. Para obter mais informações sobre a proteção de dados na Europa, consulte a postagem do blog AWS LGPD e Modelo de Responsabilidade Compartilhada no AWS Blog de Segurança.

Para fins de proteção de dados, recomendamos que você proteja Conta da AWS as credenciais e configure usuários individuais com AWS IAM Identity Center ou AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use uma autenticação multifator (MFA) com cada conta.
- Use SSL/TLS para se comunicar com os recursos. AWS Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Configure a API e o registro de atividades do usuário com AWS CloudTrail.
- Use soluções de AWS criptografia, juntamente com todos os controles de segurança padrão Serviços da AWS.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados sigilosos armazenados no Amazon S3.
- Se você precisar de módulos criptográficos validados pelo FIPS 140-2 ao acessar AWS por meio de uma interface de linha de comando ou de uma API, use um endpoint FIPS. Para ter mais informações sobre endpoints do FIPS disponíveis, consulte Federal Information Processing Standard (FIPS) 140-2.

É altamente recomendável que nunca sejam colocadas informações de identificação confidenciais, como endereços de e-mail dos seus clientes, em marcações ou campos de formato livre, como um campo Nome. Isso inclui quando você trabalha com o Neptune ou Serviços da AWS outro usando o console, a API AWS CLI ou os SDKs. AWS Quaisquer dados inseridos em tags ou campos de texto

de formato livre usados para nomes podem ser usados para logs de faturamento ou de diagnóstico. Se você fornecer um URL para um servidor externo, recomendemos fortemente que não sejam incluídas informações de credenciais no URL para validar a solicitação a esse servidor.

> **Important**
>
> O TLS 1.3 só é compatível com a versão 1.3.2.0 e superior do motor Neptune.

Você usa chamadas de API AWS publicadas para gerenciar o Neptune por meio da rede. Os clientes devem oferecer compatibilidade com Transport Layer Security (TLS) 1.2 ou posterior usando pacotes de criptografia fortes, como descrito em Criptografia em trânsito. A maioria dos sistemas modernos, como Java 7 e versões posteriores, comporta esses modos.

As seções a seguir descrevem com mais detalhes como os dados do Neptune são protegidos.

Tópicos

- Cada cluster de banco de dados do Amazon Neptune reside em uma Amazon VPC
- Criptografia em trânsito: estabelecer conexão com o Neptune Using SSL/HTTPS usando SSL/HTTPS
- Criptografia de recursos em repouso do Neptune

Cada cluster de banco de dados do Amazon Neptune reside em uma Amazon VPC

Um cluster de banco de dados do Amazon Neptune só pode ser criado em uma Amazon Virtual Private Cloud (Amazon VPC) e seus endpoints só são acessíveis dentro dessa VPC, em geral, por uma instância do Amazon Elastic Compute Cloud (Amazon EC2) em execução nessa VPC.

É possível proteger os dados do Neptune limitando o acesso à VPC em que o cluster de banco de dados do Neptune está localizado, conforme descrito em Conectar-se ao grafo do Amazon Neptune.

Criptografia em trânsito: estabelecer conexão com o Neptune Using SSL/HTTPS usando SSL/HTTPS

A partir da versão 1.0.4.0 do mecanismo, o Amazon Neptune só permite conexões Secure Sockets Layer (SSL) por meio de HTTPS para qualquer instância ou endpoint de cluster.

O Neptune requer pelo menos a versão 1.2 do TLS, usando os seguintes conjuntos de criptografia fortes:

- `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256`

A partir da versão 1.3.2.0 do Neptune Engine, o Neptune oferece suporte à versão 1.3 do TLS usando os seguintes conjuntos de criptografia:

- TLS_AES_128_GCM_SHA256
- TLS_AES_256_GCM_SHA384

Mesmo quando as conexões HTTP são permitidas em versões anteriores do mecanismo, qualquer cluster de banco de dados que use um novo grupo de parâmetros de cluster de banco de dados precisa usar SSL por padrão. Para proteger os dados, os endpoints do Neptune na versão *`1.0.4.0`* do mecanismo e posterior são compatíveis apenas com solicitações HTTPS. Consulte Usar o endpoint REST HTTP para conectar-se a uma instância de banco de dados do Neptune Para mais informações.

O Neptune fornece automaticamente certificados SSL para instâncias de banco de dados do Neptune. Você não precisa solicitar nenhum certificado. Os certificados são fornecidos ao criar uma nova instância.

O Neptune atribui um único certificado SSL curinga às instâncias em sua conta para cada região. AWS O certificado fornece entradas para os endpoints do cluster, endpoints somente leitura do cluster e endpoints de instância.

Detalhes do Certificado

As seguintes entradas estão inclusas no certificado fornecido:

- Endpoint do cluster: `*.cluster-`*`a1b2c3d4wxyz`*`.`*`region`*`.neptune.amazonaws.com`
- Endpoint somente leitura: `*.cluster-ro-`*`a1b2c3d4wxyz`*`.`*`region`*`.neptune.amazonaws.com`
- Endpoints da instância: `*.`*`a1b2c3d4wxyz`*`.`*`region`*`.neptune.amazonaws.com`

Somente as entradas listadas aqui são compatíveis.

Proxy-Connections

Os certificados oferecem suporte apenas aos nomes de host listados na seção anterior.

Se estiver usando um load balancer ou um servidor de proxy (como o HAProxy), você deve usar a terminação SSL e ter o próprio certificado no servidor de proxy.

A passagem SSL não funciona porque os certificados SSL fornecidos não correspondem ao nome do host do servidor de proxy.

Certificados Root CA

Os certificados para as instâncias do Neptune são normalmente validados usando o armazenamento de confiança local do sistema operacional ou SDK (como o SDK do Java).

Se for necessário fornecer um certificado raiz manualmente, faça download do certificado CA raiz da Amazon no formato PEM do Amazon Services Trust Policy Repository.

Mais informações

Para obter mais informações sobre como se conectar aos endpoints do Neptune com o SSL, consulte the section called “Instalar o console do Gremlin” e the section called “REST HTTP”.

Criptografia de recursos em repouso do Neptune

As instâncias criptografadas do Neptune fornecem uma camada adicional de proteção de dados, ajudando a proteger os dados contra acesso não autorizado ao armazenamento subjacente. É possível usar a criptografia do Neptune para aumentar a segurança dos dados das aplicações implantadas na nuvem. Você também pode usá-lo para atender aos requisitos de conformidade para data-at-rest criptografia.

Para gerenciar as chaves usadas para criptografar e descriptografar seus recursos do Neptune, você usa ().AWS Key Management ServiceAWS KMS AWS KMS combina hardware e software seguros e de alta disponibilidade para fornecer um sistema de gerenciamento de chaves dimensionado para a nuvem. Usando AWS KMS, você pode criar chaves de criptografia e definir as políticas que controlam como essas chaves podem ser usadas. AWS KMS suporta AWS CloudTrail, para que você possa auditar o uso das chaves para verificar se as chaves estão sendo usadas adequadamente. Você pode usar suas AWS KMS chaves em combinação com o Neptune e serviços AWS compatíveis, como Amazon Simple Storage Service (Amazon S3), Amazon Elastic Block Store

(Amazon EBS) e Amazon Redshift. Para obter uma lista dos serviços que oferecem suporte AWS KMS, consulte Como AWS os serviços são usados AWS KMS no Guia do AWS Key Management Service desenvolvedor.

Todos os logs, backups e snapshots são criptografados para uma instância criptografada do Neptune.

Habilitar a criptografia para uma instância de banco de dados do Neptune

Para habilitar a criptografia para uma nova instância de banco de dados do Neptune, selecione Sim na seção Habilitar criptografia no console do Neptune. Para obter informações sobre como criar uma instância de banco de dados do Neptune, consulte Criar um cluster de banco de dados do Neptune.

Ao criar uma instância de banco de dados Neptune criptografada, você também pode fornecer AWS KMS o identificador de chave para sua chave de criptografia. Se você não especificar um identificador de AWS KMS chave, o Neptune usará sua chave de criptografia padrão do Amazon RDS `aws/rds` () para sua nova instância de banco de dados Neptune. AWS KMS cria sua chave de criptografia padrão para Neptune para sua conta. AWS Sua AWS conta tem uma chave de criptografia padrão diferente para cada AWS região.

Depois de criar uma instância de banco de dados criptografada do Neptune, não é possível alterar a chave de criptografia para essa instância. Portanto, determine os requisitos da chave de criptografia antes de criar a instância de banco de dados do Neptune criptografada.

Você pode usar o nome do recurso da Amazon (ARN) de uma chave de outra conta para criptografar uma instância de banco de dados do Neptune. Se você criar uma instância de banco de dados Neptune com a AWS mesma conta que possui AWS KMS a chave de criptografia usada para criptografar essa nova instância de banco de dados Neptune AWS KMS , o ID da chave que você passa pode AWS KMS ser o alias da chave em vez do ARN da chave.

> **Important**
>
> Se o Neptune perder o acesso à chave de criptografia de uma instância de banco de dados do Neptune, por exemplo, quando o acesso do Neptune a uma chave for revogado, a instância de banco de dados criptografada será colocada em estado terminal e só poderá ser restaurada a partir de um backup. É altamente recomendável que você sempre habilite backups para instâncias de banco de dados criptografadas do Neptune para se proteger contra a perda de dados criptografados nos bancos de dados.

Permissões de chave necessárias ao habilitar a criptografia

O usuário ou o perfil do IAM que está criando uma instância de banco de dados criptografada do Neptune deve ter pelo menos as seguintes permissões para a chave do KMS:

- `"kms:Encrypt"`
- `"kms:Decrypt"`
- `"kms:GenerateDataKey"`
- `"kms:ReEncryptTo"`
- `"kms:GenerateDataKeyWithoutPlaintext"`
- `"kms:CreateGrant"`
- `"kms:ReEncryptFrom"`
- `"kms:DescribeKey"`

Veja um exemplo de política de chave que inclui as permissões necessárias:

```
{
  "Version": "2012-10-17",
  "Id": "key-consolepolicy-3",
  "Statement": [
    {
      "Sid": "Enable Permissions for root principal",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:root"
      },
      "Action": "kms:*",
      "Resource": "*"
    },
    {
      "Sid": "Allow use of the key for Neptune",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:role/NeptuneFullAccess"
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:GenerateDataKey",
```

```
        "kms:ReEncryptTo",
        "kms:GenerateDataKeyWithoutPlaintext",
        "kms:CreateGrant",
        "kms:ReEncryptFrom",
        "kms:DescribeKey"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "rds.us-east-1.amazonaws.com"
        }
      }
    },
    {
      "Sid": "Deny use of the key for non Neptune",
      "Effect": "Deny",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:role/NeptuneFullAccess"
      },
      "Action": [
        "kms:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "kms:ViaService": "rds.us-east-1.amazonaws.com"
        }
      }
    }
  ]
}
```

- A primeira declaração desta política é opcional. Ela concede acesso à entidade principal raiz do usuário.
- A segunda declaração fornece acesso a todas as AWS KMS APIs necessárias para essa função, com escopo reduzido ao RDS Service Principal.
- A terceira declaração aumenta ainda mais a segurança ao impor que essa chave não pode ser usada por essa função em nenhum outro serviço. AWS

Você também pode reduzir ainda mais o escopo das permissões `createGrant` adicionando:

```
"Condition": {
  "Bool": {
    "kms:GrantIsForAWSResource": true
  }
}
```

Limitações da criptografia do Neptune

Existem as seguintes limitações para a criptografia de clusters do Neptune:

- Não é possível converter um cluster de banco de dados não criptografado em um criptografado.

  Além disso, é possível restaurar um snapshot de cluster de banco de dados não criptografado do para um cluster de banco de dados criptografado do . Para fazer isso, especifique uma chave de criptografia do KMS ao restaurar a partir do snapshot do cluster de banco de dados não criptografado.
- Não é possível converter uma instância de banco de dados não criptografada em uma criptografado. Você só pode habilitar a criptografia para uma instância de banco de dados ao criá-la.
- Além disso, as instâncias criptografadas de bancos de dados não podem ser modificadas para desabilitar a criptografia.
- Você não pode ter uma réplica de leitura criptografada de uma instância de banco de dados não criptografada nem uma réplica de leitura não criptografada de uma instância de banco de dados criptografada.
- Réplicas de leitura criptografadas devem ser criptografadas com a mesma chave que a instância do banco de dados de origem.

Visão geral do AWS Identity and Access Management (IAM) no Amazon Neptune

AWS Identity and Access Management (IAM) é uma ferramenta AWS service (Serviço da AWS) que ajuda o administrador a controlar com segurança o acesso aos AWS recursos. Os administradores do IAM controlam quem pode ser autenticado (fazer login) e autorizado (ter permissões) para usar recursos do Neptune. O IAM é um AWS service (Serviço da AWS) que você pode usar sem custo adicional.

Você pode usar AWS Identity and Access Management (IAM) para se autenticar em sua instância de banco de dados ou cluster de banco de dados Neptune. Quando a autenticação do banco de dados do IAM está ativada, cada solicitação deve ser assinada usando o AWS Signature Version 4.

AWS A versão 4 do Signature adiciona informações de autenticação às AWS solicitações. Para segurança, todas as solicitações para clusters de banco de dados do Neptune DB com autenticação do IAM habilitada devem ser assinadas com uma chave de acesso. Essa chave consiste em um ID de chave de acesso e uma chave de acesso secreta. A autenticação é gerenciada externamente usando políticas do IAM.

O Neptune se autentica na conexão e, WebSockets para conexões, verifica as permissões periodicamente para garantir que o usuário ainda tenha acesso.

> **Note**
>
> - A revogação, a exclusão ou a troca de credenciais associadas ao usuário do IAM não são recomendadas porque não encerram nenhuma conexão já aberta.
> - Há limites no número de WebSocket conexões simultâneas por instância do banco de dados e por quanto tempo uma conexão pode permanecer aberta. Para ter mais informações, consulte WebSockets Limites.

O uso do IAM depende de sua função

A forma como você usa AWS Identity and Access Management (IAM) difere, dependendo do trabalho que você faz em Neptune.

Usuário do serviço: se você usa o serviço do Neptune para fazer o trabalho, o administrador fornece as credenciais e as permissões necessárias para usar o plano de dados do Neptune. À medida que você precisa de mais acesso para fazer seu trabalho, entender como o acesso é gerenciado pode ajudar você a solicitar as permissões corretas ao administrador.

Administrador do serviço: se você for o responsável pelos recursos do Neptune na empresa, provavelmente terá acesso às ações de gerenciamento do Neptune, que correspondem à API de gerenciamento do Neptune. Também pode ser seu trabalho determinar quais ações de acesso a dados e recursos do Neptune de que os usuários do serviço precisam para realizar o trabalho. Assim, um administrador do IAM poderá aplicar políticas do IAM para alterar as permissões dos usuários do serviço.

Administrador do IAM: se você for administrador do IAM, precisará elaborar políticas do IAM para gerenciar o gerenciamento e o acesso aos dados ao Neptune. Para visualizar exemplos de política baseada em identidade do Neptune que podem ser usadas, consulte Usar diferentes tipos de política do IAM para controle de acesso ao Neptune.

Autenticação com identidades

A autenticação é a forma como você faz login AWS usando suas credenciais de identidade. Você deve estar autenticado (conectado AWS) como o Usuário raiz da conta da AWS, como usuário do IAM ou assumindo uma função do IAM.

Você pode entrar AWS como uma identidade federada usando credenciais fornecidas por meio de uma fonte de identidade. AWS IAM Identity Center Usuários (IAM Identity Center), a autenticação de login único da sua empresa e suas credenciais do Google ou do Facebook são exemplos de identidades federadas. Quando você faz login como identidade federada, o administrador já configurou anteriormente a federação de identidades usando perfis do IAM. Ao acessar AWS usando a federação, você está assumindo indiretamente uma função.

Dependendo do tipo de usuário que você é, você pode entrar no AWS Management Console ou no portal de AWS acesso. Para obter mais informações sobre como fazer login AWS, consulte Como fazer login Conta da AWS no Guia do Início de Sessão da AWS usuário.

Se você acessar AWS programaticamente, AWS fornece um kit de desenvolvimento de software (SDK) e uma interface de linha de comando (CLI) para assinar criptograficamente suas solicitações usando suas credenciais. Se você não usa AWS ferramentas, você mesmo deve assinar as solicitações. Para obter mais informações sobre como usar o método recomendado para assinar solicitações por conta própria, consulte Assinatura de solicitações de AWS API no Guia do usuário do IAM.

Independente do método de autenticação usado, também pode ser exigido que você forneça informações adicionais de segurança. Por exemplo, AWS recomenda que você use a autenticação multifator (MFA) para aumentar a segurança da sua conta. Para saber mais, consulte Autenticação multifator no Guia AWS IAM Identity Center do usuário e Utilizar a autenticação multifator (MFA) na AWS no Guia do usuário do IAM.

Conta da AWS usuário root

Ao criar uma Conta da AWS, você começa com uma identidade de login que tem acesso completo a todos Serviços da AWS os recursos da conta. Essa identidade é chamada de usuário Conta da AWS raiz e é acessada fazendo login com o endereço de e-mail e a senha que você usou para criar a

conta. É altamente recomendável não usar o usuário raiz para tarefas diárias. Proteja as credenciais do usuário raiz e use-as para executar as tarefas que somente ele puder executar. Para obter a lista completa das tarefas que exigem login como usuário raiz, consulte Tarefas que exigem credenciais de usuário raiz no Guia do Usuário do IAM.

Grupos e usuários do IAM

Um usuário do IAM é uma identidade dentro da sua Conta da AWS que tem permissões específicas para uma única pessoa ou aplicativo. Sempre que possível, recomendamos contar com credenciais temporárias em vez de criar usuários do IAM com credenciais de longo prazo, como senhas e chaves de acesso. No entanto, se você tiver casos de uso específicos que exijam credenciais de longo prazo com usuários do IAM, recomendamos alternar as chaves de acesso. Para obter mais informações, consulte Alterne as chaves de acesso regularmente para casos de uso que exijam credenciais de longo prazo no Guia do Usuário do IAM.

Um grupo do IAM é uma identidade que especifica uma coleção de usuários do IAM. Não é possível fazer login como um grupo. É possível usar grupos para especificar permissões para vários usuários de uma vez. Os grupos facilitam o gerenciamento de permissões para grandes conjuntos de usuários. Por exemplo, você pode ter um grupo chamado IAMAdmins e conceder a esse grupo permissões para administrar recursos do IAM.

Usuários são diferentes de perfis. Um usuário é exclusivamente associado a uma pessoa ou a uma aplicação, mas um perfil pode ser assumido por qualquer pessoa que precisar dele. Os usuários têm credenciais permanentes de longo prazo, mas os perfis fornecem credenciais temporárias. Para saber mais, consulte Quando criar um usuário do IAM (em vez de um perfil) no Guia do usuário do IAM.

Perfis do IAM

Uma função do IAM é uma identidade dentro da sua Conta da AWS que tem permissões específicas. Ele é semelhante a um usuário do IAM, mas não está associado a uma pessoa específica. Você pode assumir temporariamente uma função do IAM no AWS Management Console trocando de funções. Você pode assumir uma função chamando uma operação de AWS API AWS CLI ou usando uma URL personalizada. Para obter mais informações sobre métodos para o uso de perfis, consulte Utilizar perfis do IAM no Guia do usuário do IAM.

Funções do IAM com credenciais temporárias são úteis nas seguintes situações:

- Acesso de usuário federado: para atribuir permissões a identidades federadas, você pode criar um perfil e definir permissões para ele. Quando uma identidade federada é autenticada, essa

identidade é associada ao perfil e recebe as permissões definidas pelo mesmo. Para obter mais informações sobre perfis para federação, consulte Criar um perfil para um provedor de identidades de terceiros no Guia do Usuário do IAM. Se você usar o Centro de identidade do IAM, configure um conjunto de permissões. Para controlar o que suas identidades podem acessar após a autenticação, o Centro de identidade do IAM correlaciona o conjunto de permissões a um perfil no IAM. Para obter informações sobre conjuntos de permissões, consulte Conjuntos de permissões no Guia do usuário AWS IAM Identity Center .

- Permissões temporárias para usuários do IAM –- um usuário ou um perfil do IAM pode presumir um perfil do IAM para obter temporariamente permissões diferentes para uma tarefa específica.
- Acesso entre contas –- é possível usar um perfil do IAM para permitir que alguém (uma entidade principal confiável) em outra conta acesse recursos em sua conta. Os perfis são a principal forma de conceder acesso entre contas. No entanto, com alguns Serviços da AWS, você pode anexar uma política diretamente a um recurso (em vez de usar uma função como proxy). Para saber a diferença entre funções e políticas baseadas em recursos para acesso entre contas, consulte Acesso a recursos entre contas no IAM no Guia do usuário do IAM.
- Acesso entre serviços — Alguns Serviços da AWS usam recursos em outros Serviços da AWS. Por exemplo, quando você faz uma chamada em um serviço, é comum que esse serviço execute aplicativos no Amazon EC2 ou armazene objetos no Amazon S3. Um serviço pode fazer isso usando as permissões do principal de chamada, usando um perfil de serviço ou um perfil vinculado a serviço.
  - Sessões de acesso direto (FAS) — Quando você usa um usuário ou uma função do IAM para realizar ações AWS, você é considerado principal. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. O FAS usa as permissões do diretor chamando um AWS service (Serviço da AWS), combinadas com a solicitação AWS service (Serviço da AWS) para fazer solicitações aos serviços posteriores. As solicitações do FAS são feitas somente quando um serviço recebe uma solicitação que requer interações com outros Serviços da AWS ou com recursos para ser concluída. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer solicitações de FAS, consulte Encaminhar sessões de acesso.
  - Função de serviço: um perfil de serviço é um perfil do IAM que um serviço assume para realizar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de serviço do IAM. Para obter mais informações, consulte Criar um perfil para delegar permissões a um AWS service (Serviço da AWS) no Guia do Usuário do IAM.
  - Função vinculada ao serviço — Uma função vinculada ao serviço é um tipo de função de serviço vinculada a um. AWS service (Serviço da AWS) O serviço pode presumir a função de executar

uma ação em seu nome. As funções vinculadas ao serviço aparecem em você Conta da AWS e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não editar as permissões para funções vinculadas ao serviço.

- Aplicativos em execução no Amazon EC2 — Você pode usar uma função do IAM para gerenciar credenciais temporárias para aplicativos que estão sendo executados em uma instância do EC2 e fazendo AWS CLI solicitações de API. AWS É preferível fazer isso a armazenar chaves de acesso na instância do EC2. Para atribuir uma AWS função a uma instância do EC2 e disponibilizá-la para todos os seus aplicativos, você cria um perfil de instância anexado à instância. Um perfil de instância contém o perfil e permite que os programas em execução na instância do EC2 obtenham credenciais temporárias. Para mais informações, consulte Utilizar um perfil do IAM para conceder permissões a aplicações em execução nas instâncias do Amazon EC2 no Guia do usuário do IAM.

Para saber se deseja usar perfis do IAM, consulte Quando criar um perfil do IAM (em vez de um usuário) no Guia do usuário do IAM.

Habilitar a autenticação de banco de dados do IAM no Neptune

Por padrão, a autenticação de banco de dados do IAM é desabilitada quando você cria um cluster de banco de dados do Amazon Neptune. Você pode habilitar a autenticação de banco de dados do IAM (ou desabilitá-la novamente) usando o AWS Management Console

Para criar um cluster de banco de dados do Neptune com autenticação do IAM usando o console, siga as instruções para criar um cluster de banco de dados do Neptune em Iniciar um cluster de banco de dados do Neptune usando o AWS Management Console.

Na segunda página do processo de criação, selecione Yes (Sim) para Enable IAM DB Authentication (Habilitar a autenticação de banco de dados do IAM).

Para habilitar ou desabilitar a autenticação do IAM para um cluster ou instância de banco de dados existente

1. Faça login no AWS Management Console e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. No painel de navegação, escolha Clusters.
3. Escolha o cluster de banco de dados do Neptune que você deseja modificar e selecione Ações de cluster. Depois, selecione Modify Cluster (Modificar cluster).

4. Na seção Database options (Opções de banco de dados), em IAM DB Authentication (Autenticação de banco de dados do IAM), escolha Enable IAM DB authorization (Ativar autorização do banco de dados do IAM) ou No (Não) (para desabilitar). Depois, escolha Continue (Continuar).
5. Para aplicar as alterações imediatamente, escolha Apply immediately.
6. Escolha Modificar cluster.

Conexão e assinatura com o AWS Signature versão 4

Os recursos do Amazon Neptune que têm a autenticação de banco de dados do IAM habilitada exigem que todas as solicitações HTTP sejam assinadas AWS usando o Signature versão 4. Para obter informações gerais sobre como assinar solicitações com o AWS Signature versão 4, consulte Assinatura de solicitações AWS da API.

AWS A versão 4 do Signature é o processo para adicionar informações de autenticação às AWS solicitações. Por motivos de segurança, a maioria das solicitações AWS deve ser assinada com uma chave de acesso, que consiste em uma ID da chave de acesso e uma chave de acesso secreta.

> **Note**
>
> Se você estiver usando credenciais temporárias, elas expirarão após um intervalo especificado, incluindo o token de sessão.
> Você deve atualizar o seu token de sessão ao solicitar novas credenciais. Para obter mais informações, consulte Usando credenciais de segurança temporárias para solicitar acesso aos AWS recursos.

> **Important**
>
> O acesso ao Neptune com autenticação baseada no IAM exige que você crie solicitações HTTP e assine-as.

Como funciona o Signature versão 4

1. Você cria uma solicitação canônica.
2. Você usa a solicitação canônica e algumas outras informações para criar uma. string-to-sign

3. Você usa sua chave de acesso AWS secreta para derivar uma chave de assinatura e, em seguida, usa essa chave de assinatura e a string-to-sign para criar uma assinatura.
4. Você adiciona a assinatura resultante para a solicitação de HTTP em um cabeçalho ou como um parâmetro de string de consulta.

Quando recebe a solicitação, o Neptune executa as mesmas etapas que você executou para calcular a assinatura. Depois, o Neptune compara a assinatura calculada com a enviada com a solicitação. Se as assinaturas corresponderem, a solicitação é processada. Se as assinaturas não corresponderem, a solicitação é negada.

Para obter informações gerais sobre como assinar solicitações com o AWS Signature versão 4, consulte Processo de assinatura do Signature versão 4 no Referência geral da AWS.

As seções a seguir mostram exemplos que ilustram como enviar solicitações assinadas aos endpoints do Gremlin e do SPARQL de uma instância de banco de dados do Neptune com a autenticação do IAM habilitada.

Tópicos

Pré-requisitos no Amazon Linux EC2

Veja as instruções para instalar o Apache Maven e Java 8 em uma instância do Amazon EC2. Elas são necessárias para exemplos de autenticação do Amazon Neptune Signature versão 4.

Para instalar o Apache Maven e o Java 8 na instância do EC2

1. Conecte-se à instância do Amazon EC2 com um cliente SSH.
2. Instale o Apache Maven na instância do EC2. Primeiro, insira o seguinte para adicionar um repositório com um pacote do Maven.

   ```
   sudo wget https://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-apache-maven.repo
   ```

   Insira o seguinte para definir o número da versão para os pacotes.

   ```
   sudo sed -i s/\$releasever/6/g /etc/yum.repos.d/epel-apache-maven.repo
   ```

   Em seguida, você pode usar yum para instalar o Maven.

   ```
   sudo yum install -y apache-maven
   ```
3. As bibliotecas do Gremlin exigem o Java 8. Insira o seguinte para instalar o Java 8 na instância do EC2.

   ```
   sudo yum install java-1.8.0-devel
   ```
4. Insira o seguinte para definir o Java 8 como o tempo de execução padrão na instância do EC2.

   ```
   sudo /usr/sbin/alternatives --config java
   ```

   Quando solicitado, insira o número do Java 8.
5. Insira o seguinte para definir o Java 8 como o compilador padrão na instância do EC2.

   ```
   sudo /usr/sbin/alternatives --config javac
   ```

   Quando solicitado, insira o número do Java 8.

Usar uma ferramenta de linha de comando para enviar consultas ao cluster de banco de dados do Neptune

Ter uma ferramenta de linha de comando para enviar consultas ao cluster de banco de dados do Neptune é muito útil, conforme ilustrado em muitos dos exemplos desta documentação. A ferramenta curl é uma excelente opção para se comunicar com os endpoints do Neptune quando a autenticação do IAM não está habilitada.

No entanto, para manter os dados seguros, é melhor habilitar a autenticação do IAM.

Quando a autenticação do IAM está habilitada, cada solicitação deve ser assinada usando o Signature versão 4 (Sig4). A ferramenta de linha de comando awscurl de terceiros usa a mesma sintaxe que `curl` e pode assinar consultas usando a assinatura Sig4. A seção Utilizar o `awscurl` abaixo explica como usar `awscurl` com segurança credenciais temporárias.

Configurar uma ferramenta de linha de comando para usar HTTPS

O Neptune exige que todas as conexões usem HTTPS. Qualquer ferramenta de linha de comando, como `curl` ou `awscurl`, precisa acessar os certificados apropriados, a fim de usar HTTPS. Desde que o `curl` e o `awscurl` possam localizar os certificados adequados, eles tratam as conexões HTTPS da mesma forma que as conexões HTTP, sem precisar de parâmetros extras. Os exemplos desta documentação se baseiam nesse cenário.

Para saber como obter esses certificados e como formatá-los corretamente em um armazenamento de certificados (CA) que o `curl` possa usar, consulte SSL Certificate Verification na documentação do `curl`.

Depois, você pode especificar o local desse armazenamento de certificados CA usando a variável de ambiente `CURL_CA_BUNDLE`. No Windows, o `curl` os procura automaticamente em um arquivo chamado `curl-ca-bundle.crt`. Ele procura primeiro no mesmo diretório `curl.exe` e, em seguida, em outros lugares no caminho. Para obter mais informações, consulte SSL Certificate Verification.

Usar **`awscurl`** com credenciais temporárias para se conectar com segurança a um cluster de banco de dados com a autenticação do IAM habilitada

A ferramenta awscurl usa a mesma sintaxe que `curl`, mas também precisa de informações adicionais:

- **`--access_key`**: uma chave de acesso válida. Se não for fornecida usando esse parâmetro, ela deverá ser fornecida na variável de ambiente `AWS_ACCESS_KEY_ID` ou em um arquivo de configuração.
- **`--secret_key`**: a chave de acesso secreta que corresponde à chave de acesso. Se não for fornecida usando esse parâmetro, ela deverá ser fornecida na variável de ambiente `AWS_SECRET_ACCESS_KEY` ou em um arquivo de configuração.
- **`--security_token`**: um token de sessão válido. Se não for fornecida usando esse parâmetro, ela deverá ser fornecida na variável de ambiente `AWS_SECURITY_TOKEN` ou em um arquivo de configuração.

No passado, era uma prática comum usar credenciais persistentes com `awscurl`, como credenciais de usuário do IAM ou até mesmo credenciais raiz, mas isso não é recomendado. Em vez disso, gere credenciais temporárias usando uma das APIs do AWS Security Token Service (STS) ou um de seus wrappers da AWS CLI.

É melhor colocar os valores `AccessKeyId`, `SecretAccessKey` e `SessionToken` são gerados pela chamada STS nas variáveis de ambiente apropriadas em sua sessão de shell e não em um arquivo de configuração. Então, quando o shell for encerrado, as credenciais serão automaticamente descartadas, diferente de um arquivo de configuração. Da mesma forma, não solicite para as credenciais temporárias uma duração maior do que você provavelmente precisará.

O exemplo a seguir mostra as etapas que você pode executar em um shell Linux para obter credenciais temporárias válidas por meia hora usando sts assume-role e, depois, colocá-las em variáveis de ambiente em que `awscurl` pode encontrá-las:

```
aws sts assume-role \
    --duration-seconds 1800 \
    --role-arn "arn:aws:iam::(account-id):role/(rolename)" \
    --role-session-name AWSCLI-Session > $output
AccessKeyId=$(cat $output | jq '.Credentials''.AccessKeyId')
SecretAccessKey=$(cat $output | jq '.Credentials''.SecretAccessKey')
SessionToken=$(cat $output | jq '.Credentials''.SessionToken')

export AWS_ACCESS_KEY_ID=$AccessKeyId
export AWS_SECRET_ACCESS_KEY=$SecretAccessKey
export AWS_SESSION_TOKEN=$SessionToken
```

Depois, você pode usar `awscurl` para fazer uma solicitação assinada para o cluster de banco de dados mais ou menos desta forma:

```
awscurl (your cluster endpoint):8182/status \
    --region us-east-1 \
    --service neptune-db
```

Estabelecer conexão com o Neptune usando o console do Gremlin com a assinatura do Signature versão 4

A forma como você se conecta ao Amazon Neptune usando o console Gremlin com a autenticação Signature versão 4 depende se você está TinkerPop usando uma `3.4.11` versão ou superior, ou uma versão anterior. Nos dois casos, os seguintes pré-requisitos são necessários:

- É necessário ter as credenciais do IAM necessárias para assinar as solicitações. Consulte Uso da cadeia de fornecedores de credenciais padrão no Guia do AWS SDK for Java desenvolvedor.
- Você deve ter instalado uma versão do console do Gremlin que seja compatível com a versão do mecanismo do Neptune que está sendo usada pelo cluster de banco de dados.

Se você estiver usando credenciais temporárias, elas vão expirar após um intervalo especificado, assim como o token da sessão. Portanto, é necessário atualizar o token da sessão ao solicitar novas credenciais. Consulte Uso de credenciais de segurança temporárias para solicitar acesso aos AWS recursos no Guia do usuário do IAM.

Para obter ajuda na conexão usando SSL/TLS, consulte Configuração de SSL/TLS.

Usando TinkerPop 3.4.11 ou superior para se conectar a Neptune com assinatura Sig4

Com a TinkerPop versão 3.4.11 ou superior, você usará`handshakeInterceptor()`, que fornece uma maneira de conectar um assinante Sigv4 à conexão estabelecida pelo comando. `:remote` Assim como na abordagem utilizada para Java, ela exige que você configure o objeto `Cluster` manualmente e depois o transmita ao comando `:remote`.

Observe que isso é bem diferente da situação típica em que o comando `:remote` usa um arquivo de configuração para estabelecer a conexão. A abordagem do arquivo de configuração não funcionará porque `handshakeInterceptor()` deve ser definida de modo programático e não pode carregar a configuração por um arquivo.

Conecte o console Gremlin (TinkerPop 3.4.11 e superior) com a assinatura Sig4

1. Inicie o console do Gremlin:

```
$ bin/gremlin.sh
```

2. No prompt `gremlin>`, instale a biblioteca `amazon-neptune-sigv4-signer` (é necessário realizar esse procedimento apenas uma vez para o console):

```
:install com.amazonaws amazon-neptune-sigv4-signer 2.4.0
```

Se você encontrar problemas com essa etapa, pode ser útil consultar a TinkerPop documentação sobre a configuração do Grape.

> **Note**
>
> Se você estiver usando um proxy HTTP, poderá encontrar erros nessa etapa em que o comando `:install` não é concluído. Para corrigir esse problema, execute os comandos a seguir para informar o console sobre o proxy:
>
> ```
> System.setProperty("https.proxyHost", "(the proxy IP address)")
> System.setProperty("https.proxyPort", "(the proxy port)")
> ```

3. Importe a classe necessária para lidar com o login em `handshakeInterceptor()`:

```
:import com.amazonaws.auth.DefaultAWSCredentialsProviderChain
:import com.amazonaws.neptune.auth.NeptuneNettyHttpSigV4Signer
```

4. Se você estiver usando credenciais temporárias, também precisará fornecer o token de sessão da seguinte forma:

```
System.setProperty("aws.sessionToken","(your session token)")
```

5. Se você não estabeleceu as credenciais da conta de outra forma, poderá atribuí-las da seguinte maneira:

```
System.setProperty("aws.accessKeyId","(your access key)")
System.setProperty("aws.secretKey","(your secret key)")
```

6. Crie manualmente o objeto `Cluster` para conectar-se ao Neptune:

```
cluster = Cluster.build("(host name)")  \
                .enableSsl(true) \
                .handshakeInterceptor { r ->  \
                  def sigV4Signer = new NeptuneNettyHttpSigV4Signer("(Amazon
 region)", \
                                        new DefaultAWSCredentialsProviderChain()); \
                  sigV4Signer.signRequest(r); \
                  return r; } \
                .create()
```

Para obter ajuda para encontrar o nome do host de uma instância de banco de dados do Neptune, consulte Conectar-se a endpoints do Amazon Neptune.

7. Estabeleça a conexão `:remote` usando o nome da variável do objeto `Cluster` na etapa anterior:

```
:remote connect tinkerpop.server cluster
```

8. Insira o comando a seguir para alternar para modo remoto. Isso envia todas as consultas do Gremlin para a conexão remota:

```
:remote console
```

Usando uma versão TinkerPop anterior à 3.4.11 para se conectar ao Netuno com a assinatura Sig4

Com a TinkerPop versão 3.4.10 ou inferior, use a `amazon-neptune-gremlin-java-sigv4` biblioteca fornecida pelo Neptune para conectar o console ao Neptune com a assinatura Sigv4, conforme descrito abaixo:

Conecte o console Gremlin (TinkerPop versões anteriores à 3.4.11) com a assinatura Sig4

1. Inicie o console do Gremlin:

```
$ bin/gremlin.sh
```

2. No prompt `gremlin>`, instale a biblioteca `amazon-neptune-sigv4-signer` (é necessário realizar esse procedimento apenas uma vez para o console):

```
:install com.amazonaws amazon-neptune-sigv4-signer 2.4.0
```

> **Note**
>
> Se você estiver usando um proxy HTTP, poderá encontrar erros nessa etapa em que o comando `:install` não é concluído. Para corrigir esse problema, execute os comandos a seguir para informar o console sobre o proxy:
>
> ```
> System.setProperty("https.proxyHost", "(the proxy IP address)")
> System.setProperty("https.proxyPort", "(the proxy port)")
> ```
>
> Também pode ser útil consultar a TinkerPop documentação sobre a configuração do Grape.

3. No subdiretório `conf` do diretório extraído, crie um arquivo chamado `neptune-remote.yaml`.

   Se você usou o AWS CloudFormation modelo para criar seu cluster de banco de dados Neptune, `neptune-remote.yaml` um arquivo já existirá. Nesse caso, basta editar o arquivo existente para incluir a configuração do canalizador mostrada abaixo.

   Caso contrário, copie o texto a seguir no arquivo, substituindo *(nome do host)* pelo nome do host ou o endereço IP da instância de banco de dados do Neptune. Observe que os colchetes ([]) ao lado do nome do host são obrigatórios.

   ```
   hosts: [(host name)]
   port: 8182
   connectionPool: {
     channelizer: org.apache.tinkerpop.gremlin.driver.SigV4WebSocketChannelizer,
     enableSsl: true
   }
   serializer: { className:
    org.apache.tinkerpop.gremlin.driver.ser.GraphBinaryMessageSerializerV1,
                 config: { serializeResultToString: true }}
   ```

4.

> **Important**
>
> É necessário fornecer credenciais do IAM para assinar as solicitações. Digite os comandos a seguir para definir as credenciais como variáveis de ambiente, substituindo os itens relevantes por suas credenciais.
>
> ```
> export AWS_ACCESS_KEY_ID=access_key_id
> export AWS_SECRET_ACCESS_KEY=secret_access_key
> export SERVICE_REGION=us-east-1 or us-east-2 or us-west-1 or us-west-2 or
>  ca-central-1 or
>                       sa-east-1 or eu-north-1 or eu-west-1 or eu-west-2 or
>  eu-west-3 or eu-central-1 or me-south-1 or
>                       me-central-1 or il-central-1 or af-south-1 or
>  ap-east-1 or ap-northeast-1 or ap-northeast-2 or ap-southeast-1 or ap-
> southeast-2 or ap-south-1 or
>                       cn-north-1 or cn-northwest-1 or
>                       us-gov-east-1 or us-gov-west-1
> ```
>
> O Neptune versão 4 Signer usa a cadeia de fornecedores de credenciais padrão. Para conhecer métodos adicionais de fornecimento de credenciais, consulte Using the Default Credential Provider Chain no Guia do desenvolvedor do AWS SDK for Java .
> A variável `SERVICE_REGION` é necessária, mesmo ao usar um arquivo de credenciais.

5. Estabeleça a conexão `:remote` usando o arquivo `.yaml`:

```
:remote connect tinkerpop.server conf/neptune-remote.yaml
```

6. Digite o seguinte comando para alternar para o modo remoto, que envia todas as consultas do Gremlin para a conexão remota:

```
:remote console
```

Estabelecer conexão com o Neptune usando Java e Gremlin com a assinatura do Signature versão 4

Usando TinkerPop 3.4.11 ou superior para se conectar a Neptune com assinatura Sig4

Aqui está um exemplo de como se conectar ao Neptune usando a API Gremlin Java com assinatura Sig4 ao usar 3.4.11 ou superior (pressupõe conhecimento geral sobre TinkerPop o uso do Maven). Primeiro, defina as dependências como parte do arquivo `pom.xml`:

```
<dependency>
  <groupId>com.amazonaws</groupId>
  <artifactId>amazon-neptune-sigv4-signer</artifactId>
  <version>2.4.0</version>
</dependency>
```

Depois, use o código da seguinte maneira:

```
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.neptune.auth.NeptuneNettyHttpSigV4Signer;
import com.amazonaws.neptune.auth.NeptuneSigV4SignerException;

 ...

System.setProperty("aws.accessKeyId","your-access-key");
System.setProperty("aws.secretKey","your-secret-key");

 ...

Cluster cluster = Cluster.build((your cluster))
                 .enableSsl(true)
                 .handshakeInterceptor( r ->
                  {
                    try {
                      NeptuneNettyHttpSigV4Signer sigV4Signer =
                        new NeptuneNettyHttpSigV4Signer("(your region)", new
 DefaultAWSCredentialsProviderChain());
                      sigV4Signer.signRequest(r);
                    } catch (NeptuneSigV4SignerException e) {
                      throw new RuntimeException("Exception occurred while signing the
 request", e);
                    }
                    return r;
```

```
                  }
                 ).create();
try {
  Client client = cluster.connect();
  client.submit("g.V().has('code','IAD')").all().get();
} catch (Exception e) {
  throw new RuntimeException("Exception occurred while connecting to cluster", e);
}
```

> **Note**
>
> Se você estiver realizando a atualização do `3.4.11`, remova as referências à biblioteca `amazon-neptune-gremlin-java-sigv4`. Não é mais necessária ao usar `handshakeInterceptor()` conforme mostrado no exemplo acima. Não tente usar o `handshakeInterceptor()` com o canalizador (`SigV4WebSocketChannelizer.class`), pois isso gerará erros.

Usando uma versão TinkerPop anterior à 3.4.11 para se conectar ao Netuno com a assinatura Sig4

TinkerPop as versões anteriores `3.4.11` não tinham suporte para a `handshakeInterceptor()` configuração mostrada na seção anterior e, portanto, devem contar com o `amazon-neptune-gremlin-java-sigv4` pacote. Essa é uma biblioteca Neptune que contém a classe, que substitui `SigV4WebSocketChannelizer` o Channelizer TinkerPop padrão por uma que pode injetar automaticamente uma assinatura SigV4. Sempre que possível, atualize para TinkerPop 3.4.11 ou superior, porque a `amazon-neptune-gremlin-java-sigv4` biblioteca está obsoleta.

Aqui está um exemplo de como se conectar ao Neptune usando a API Gremlin Java com assinatura Sig4 ao usar versões anteriores à 3.4.11 (pressupõe conhecimento geral sobre como TinkerPop usar o Maven).

Primeiro, defina as dependências como parte do arquivo `pom.xml`:

```
<dependency>
  <groupId>com.amazonaws</groupId>
  <artifactId>amazon-neptune-gremlin-java-sigv4</artifactId>
  <version>2.4.0</version>
</dependency>
```

A dependência acima incluirá a versão `3.4.10` do driver do Gremlin. Embora seja possível usar versões mais recentes do driver do Gremlin (até `3.4.13`), uma atualização do driver após a 3.4.10 deve incluir uma alteração para usar o modelo `handshakeInterceptor()` descrito acima.

O objeto `gremlin-driver` do cluster deve então ser configurado da seguinte forma no código Java:

```
import org.apache.tinkerpop.gremlin.driver.SigV4WebSocketChannelizer;

 ...

Cluster cluster = Cluster.build(your cluster)
                        .enableSsl(true)
                        .channelizer(SigV4WebSocketChannelizer.class)
                        .create();
Client client = cluster.connect();
client.submit("g.V().has('code','IAD')").all().get();
```

Estabelecer conexão com o Neptune Using Java usando Java e SPARQL com assinatura do Signature versão 4 (RDF4J e Jena)

Esta seção mostra como se conectar ao Neptune usando RDF4J ou Apache Jena com autenticação do Signature versão 4.

Pré-requisitos

- Java 8 ou posterior.
- Apache Maven 3.3 ou posterior.

  Para obter informações sobre como instalar esses pré-requisitos em uma instância do EC2 executando o Amazon Linux, consulte Pré-requisitos no Amazon Linux EC2.
- Credenciais do IAM para assinar as solicitações. Para obter mais informações, consulte Using the Default Credential Provider Chain no Guia do desenvolvedor do AWS SDK for Java .

> **Note**
>
> Se você estiver usando credenciais temporárias, elas expirarão após um intervalo especificado, incluindo o token de sessão.

> Você deve atualizar o seu token de sessão ao solicitar novas credenciais. Para obter mais informações, consulte Como usar credenciais de segurança temporárias para solicitar acesso a AWS recursos no Guia do usuário do IAM.

- Defina a variável `SERVICE_REGION` como um dos seguintes, indicando a região da instância de banco de dados do Neptune:
  - Leste dos EUA (Norte da Virgínia): `us-east-1`
  - Leste dos EUA (Ohio): `us-east-2`
  - Oeste dos EUA (N. da Califórnia): `us-west-1`
  - Oeste dos EUA (Oregon): `us-west-2`
  - Canadá (Central): `ca-central-1`
  - América do Sul (São Paulo): `sa-east-1`
  - Europa (Estocolmo): `eu-north-1`
  - Europa (Irlanda): `eu-west-1`
  - Europa (Londres): `eu-west-2`
  - Europa (Paris): `eu-west-3`
  - Europa (Frankfurt): `eu-central-1`
  - Oriente Médio (Bahrein): `me-south-1`
  - Oriente Médio (Emirados Árabes Unidos): `me-central-1`
  - Israel (Tel Aviv): `il-central-1`
  - África (Cidade do Cabo): `af-south-1`
  - Ásia-Pacífico (Hong Kong): `ap-east-1`
  - Ásia-Pacífico (Tóquio): `ap-northeast-1`
  - Ásia-Pacífico (Seul): `ap-northeast-2`
  - Ásia-Pacífico (Osaka): `ap-northeast-3`
  - Ásia-Pacífico (Singapura): `ap-southeast-1`
  - Ásia-Pacífico (Sydney): `ap-southeast-2`
  - Ásia-Pacífico (Mumbai): `ap-south-1`
  - China (Pequim): `cn-north-1`
  - China (Ningxia): `cn-northwest-1`
  - AWS GovCloud (Oeste dos EUA): `us-gov-west-1`

- AWS GovCloud (Leste dos EUA): `us-gov-east-1`

Como se conectar ao Neptune usando RDF4J ou Apache Jena com a assinatura do Signature 4

1. Clone o repositório de amostra de. GitHub

```
git clone https://github.com/aws/amazon-neptune-sparql-java-sigv4.git
```

2. Altere no diretório clonado.

```
cd amazon-neptune-sparql-java-sigv4
```

3. Obtenha a versão mais recente do projeto marcando a ramificação com a última tag.

```
git checkout $(git describe --tags `git rev-list --tags --max-count=1`)
```

4. Digite um dos comandos a seguir para compilar e executar o código de exemplo.

   Substitua *your-neptune-endpoint* pelo nome de host ou endereço IP da instância de banco de dados do Neptune. A porta padrão é 8182.

   > **Note**
   >
   > Para obter informações sobre como localizar o nome do host da instância de banco de dados do Neptune, consulte a seção Conectar-se a endpoints do Amazon Neptune.

   Eclipse RDF4J

   Digite o seguinte para executar o exemplo RDF4J.

```
mvn compile exec:java \
    -Dexec.mainClass="com.amazonaws.neptune.client.rdf4j.NeptuneRdf4JSigV4Example"
 \
    -Dexec.args="https://your-neptune-endpoint:port"
```

   Apache Jena

   Digite o seguinte para executar o Apache Jena de exemplo.

```
mvn compile exec:java \
    -Dexec.mainClass="com.amazonaws.neptune.client.jena.NeptuneJenaSigV4Example" \
    -Dexec.args="https://your-neptune-endpoint:port"
```

5. Para visualizar o código-fonte do exemplo, consulte os exemplos no diretório `src/main/java/com/amazonaws/neptune/client/`.

Para usar o driver de assinatura SigV4 no seu próprio aplicativo Java, adicione o pacote Maven `amazon-neptune-sigv4-signer` à seção `<dependencies>` do `pom.xml`. Recomendamos que você use os exemplos como um ponto de partida.

Estabelecer conexão com o Neptune usando SPARQL e Node.js com assinatura do Signature versão 4

Consulta usando a assinatura Signature V4 e o AWS SDK para Javascript V3

Aqui está um exemplo de como se conectar ao Neptune SPARQL usando o Node.js com a autenticação Signature versão 4 e AWS o SDK para Javascript V3:

```
const { HttpRequest }  = require('@smithy/protocol-http');
const { fromNodeProviderChain } = require('@aws-sdk/credential-providers');
const { SignatureV4 } = require('@smithy/signature-v4');
const { Sha256 } = require('@aws-crypto/sha256-universal');
const https = require('https');

var region = 'us-west-2'; // e.g. us-west-1
var neptune_endpoint = 'your-Neptune-cluster-endpoint';  // like: 'cluster-
id.region.neptune.amazonaws.com'
var query = `query=PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX class: <http://aws.amazon.com/neptune/csv2rdf/class/>
PREFIX resource: <http://aws.amazon.com/neptune/csv2rdf/resource/>
PREFIX prop: <http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/>
PREFIX objprop: <http://aws.amazon.com/neptune/csv2rdf/objectProperty/>

SELECT ?movies ?title WHERE {
  ?jel prop:name "James Earl Jones" .
  ?movies ?p2 ?jel .
  ?movies prop:title ?title
} LIMIT 10`;
```

```
runQuery(query);

function runQuery(q) {
  var request = new HttpRequest({
    hostname: neptune_endpoint,
    port: 8182,
    path: 'sparql',
    body: encodeURI(query),
    headers: {
      'Content-Type': 'application/x-www-form-urlencoded',
      'host': neptune_endpoint + ':8182',
    },
    method: 'POST',
  });

  const credentialProvider = fromNodeProviderChain();
  let credentials = credentialProvider();
  credentials.then(
    (cred)=>{
      var signer = new SignatureV4({credentials: cred, region: region, sha256: Sha256,
 service: 'neptune-db'});
      signer.sign(request).then(
        (req)=>{
          var responseBody = '';
          var sendreq = https.request(
            {
              host: req.hostname,
              port: req.port,
              path: req.path,
              method: req.method,
              headers: req.headers,
            },
          (res) => {
            res.on('data', (chunk) => { responseBody += chunk; });
            res.on('end', () => {
                console.log(JSON.parse(responseBody));
            });
          });
          sendreq.write(req.body);
          sendreq.end();
        }
      );
    },
    (err)=>{
```

```
      console.error(err);
    }
  );
}
```

Consulta usando a assinatura Signature V4 e o AWS SDK para Javascript V2

Aqui está um exemplo de como se conectar ao Neptune SPARQL usando o Node.js com a autenticação Signature versão 4 e AWS o SDK para Javascript V2:

```
var AWS = require('aws-sdk');

var region = 'us-west-2'; // e.g. us-west-1
var neptune_endpoint = 'your-Neptune-cluster-endpoint';  // like: 'cluster-
id.region.neptune.amazonaws.com'
var query = `query=PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX class: <http://aws.amazon.com/neptune/csv2rdf/class/>
PREFIX resource: <http://aws.amazon.com/neptune/csv2rdf/resource/>
PREFIX prop: <http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/>
PREFIX objprop: <http://aws.amazon.com/neptune/csv2rdf/objectProperty/>

SELECT ?movies ?title WHERE {
    ?jel prop:name "James Earl Jones" .
    ?movies ?p2 ?jel .
    ?movies prop:title ?title
} LIMIT 10`;

runQuery(query);

function runQuery(q) {

    var endpoint = new AWS.Endpoint(neptune_endpoint);
    endpoint.port = 8182;
    var request = new AWS.HttpRequest(endpoint, region);
    request.path += 'sparql';
    request.body = encodeURI(query);
    request.headers['Content-Type'] = 'application/x-www-form-urlencoded';
    request.headers['host'] = neptune_endpoint;
    request.method = 'POST';

    var credentials = new AWS.CredentialProviderChain();
    credentials.resolve((err, cred)=>{
        var signer = new AWS.Signers.V4(request, 'neptune-db');
```

```
        signer.addAuthorization(cred, new Date());
    });

    var client = new AWS.HttpClient();
    client.handleRequest(request, null, function(response) {
        console.log(response.statusCode + ' ' + response.statusMessage);
        var responseBody = '';
        response.on('data', function (chunk) {
            responseBody += chunk;
        });
        response.on('end', function (chunk) {
            console.log('Response body: ' + responseBody);
        });
    }, function(error) {
        console.log('Error: ' + error);
    });
}
```

Exemplo: Estabelecer conexão com o Neptune usando Python com a assinatura do Signature versão 4

Esta seção mostra um exemplo de programa escrito em Python que ilustra como trabalhar com o Signature versão 4 no Amazon Neptune. Este exemplo é baseado nos exemplos da seção Processo de assinatura do Signature versão 4 na Referência geral da Amazon Web Services.

Para trabalhar com esse programa de exemplo, você precisa do seguinte:

- O Python 3.x. instalado em seu computador, que você pode obter no site da Python. Esses programas foram testados usando Python 3.6.
- A biblioteca de solicitações do Python, que é usada no script de exemplo para fazer solicitações da web. Uma forma conveniente para instalar pacotes do Python é usar `pip`, que obtém pacotes do site de índices de pacotes da Python. Depois, você pode instalar o `requests` executando `pip install requests` na linha de comando.
- Uma chave de acesso (ID da chave de acesso e chave de acesso secreta) em variáveis de ambiente chamadas `AWS_ACCESS_KEY_ID` e `AWS_SECRET_ACCESS_KEY`. Como melhores práticas, recomendamos que você não incorpore credenciais no código. Para obter mais informações, consulte Melhores práticas para contas da AWS no Guia de referência do AWS Account Management .

A região do cluster de banco de dados do Neptune na variável de ambiente denominada `SERVICE_REGION`.

Se estiver usando credenciais temporárias, você deve especificar `AWS_SESSION_TOKEN` bem como `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, e `SERVICE_REGION`.

> **Note**
>
> Se você estiver usando credenciais temporárias, elas expirarão após um intervalo especificado, incluindo o token de sessão.
> Você deve atualizar o seu token de sessão ao solicitar novas credenciais. Para obter mais informações, consulte Using Temporary Security Credentials to Request Access to AWS Resources.

O exemplo a seguir mostra como fazer solicitações assinadas ao Neptune usando o Python. A solicitação faz uma solicitação GET ou POST. As informações de autenticação são transmitidas usando o cabeçalho de solicitação `Authorization`.

Esse exemplo também funciona como uma AWS Lambda função. Para ter mais informações, consulte the section called “Configuração de Lambda”.

Como fazer solicitações assinadas aos endpoints do Gremlin e do SPARQL no Neptune

1. Crie um novo arquivo denominado `neptunesigv4.py` e abra-o em um editor de texto.
2. Copie e cole o seguinte código no arquivo `neptunesigv4.py`.

```
# Amazon Neptune version 4 signing example (version v3)

# The following script requires python 3.6+
#    (sudo yum install python36 python36-virtualenv python36-pip)
# => the reason is that we're using urllib.parse() to manually encode URL
#    parameters: the problem here is that SIGV4 encoding requires whitespaces
#    to be encoded as %20 rather than not or using '+', as done by previous/
#    default versions of the library.


# See: https://docs.aws.amazon.com/general/latest/gr/sigv4_signing.html
import sys, datetime, hashlib, hmac
import requests  # pip3 install requests
```

```
import urllib
import os
import json
from botocore.auth import SigV4Auth
from botocore.awsrequest import AWSRequest
from botocore.credentials import ReadOnlyCredentials
from types import SimpleNamespace
from argparse import RawTextHelpFormatter
from argparse import ArgumentParser

# Configuration. https is required.
protocol = 'https'

# The following lines enable debugging at httplib level (requests->urllib3-
>http.client)
# You will see the REQUEST, including HEADERS and DATA, and RESPONSE with HEADERS
 but without DATA.
#
# The only thing missing will be the response.body which is not logged.
#
# import logging
# from http.client import HTTPConnection
# HTTPConnection.debuglevel = 1
# logging.basicConfig()
# logging.getLogger().setLevel(logging.DEBUG)
# requests_log = logging.getLogger("requests.packages.urllib3")
# requests_log.setLevel(logging.DEBUG)
# requests_log.propagate = True


# Read AWS access key from env. variables. Best practice is NOT
# to embed credentials in code.
access_key = os.getenv('AWS_ACCESS_KEY_ID', '')
secret_key = os.getenv('AWS_SECRET_ACCESS_KEY', '')
region = os.getenv('SERVICE_REGION', '')

# AWS_SESSION_TOKEN is optional environment variable. Specify a session token only
 if you are using temporary
# security credentials.
session_token = os.getenv('AWS_SESSION_TOKEN', '')

### Note same script can be used for AWS Lambda (runtime = python3.6).
## Steps to use this python script for AWS Lambda
```

```
# 1. AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY and AWS_SESSION_TOKEN and AWS_REGION
 variables are already part of Lambda's Execution environment
#    No need to set them up explicitly.
# 3. Create Lambda deployment package https://docs.aws.amazon.com/lambda/latest/dg/
lambda-python-how-to-create-deployment-package.html
# 4. Create a Lambda function in the same VPC and assign an IAM role with neptune
 access

def lambda_handler(event, context):
    # sample_test_input = {
    #     "host": "END_POINT:8182",
    #     "method": "GET",
    #     "query_type": "gremlin",
    #     "query": "g.V().count()"
    # }

    # Lambda uses AWS_REGION instead of SERVICE_REGION
    global region
    region = os.getenv('AWS_REGION', '')

    host = event['host']
    method = event['method']
    query_type = event['query_type']
    query =  event['query']

    return make_signed_request(host, method, query_type, query)

def validate_input(method, query_type):
    # Supporting GET and POST for now:
    if (method != 'GET' and method != 'POST'):
        print('First parameter must be "GET" or "POST", but is "' + method + '".')
        sys.exit()

    # SPARQL UPDATE requires POST
    if (method == 'GET' and query_type == 'sparqlupdate'):
        print('SPARQL UPDATE is not supported in GET mode. Please choose POST.')
        sys.exit()

def get_canonical_uri_and_payload(query_type, query, method):
    # Set the stack and payload depending on query_type.
    if (query_type == 'sparql'):
        canonical_uri = '/sparql/'
        payload = {'query': query}
```

```
    elif (query_type == 'sparqlupdate'):
        canonical_uri = '/sparql/'
        payload = {'update': query}

    elif (query_type == 'gremlin'):
        canonical_uri = '/gremlin/'
        payload = {'gremlin': query}
        if (method == 'POST'):
            payload = json.dumps(payload)

    elif (query_type == 'openCypher'):
        canonical_uri = '/openCypher/'
        payload = {'query': query}

    elif (query_type == "loader"):
        canonical_uri = "/loader/"
        payload = query

    elif (query_type == "status"):
        canonical_uri = "/status/"
        payload = {}

    elif (query_type == "gremlin/status"):
        canonical_uri = "/gremlin/status/"
        payload = {}

    elif (query_type == "openCypher/status"):
        canonical_uri = "/openCypher/status/"
        payload = {}

    elif (query_type == "sparql/status"):
        canonical_uri = "/sparql/status/"
        payload = {}

    else:
        print(
            'Third parameter should be from ["gremlin", "sparql", "sparqlupdate",
 "loader", "status] but is "' + query_type + '".')
        sys.exit()
    ## return output as tuple
    return canonical_uri, payload

def make_signed_request(host, method, query_type, query):
    service = 'neptune-db'
```

```
    endpoint = protocol + '://' + host

    print()
    print('+++++ USER INPUT +++++')
    print('host = ' + host)
    print('method = ' + method)
    print('query_type = ' + query_type)
    print('query = ' + query)

    # validate input
    validate_input(method, query_type)

    # get canonical_uri and payload
    canonical_uri, payload = get_canonical_uri_and_payload(query_type, query,
method)

    # assign payload to data or params
    data = payload if method == 'POST' else None
    params = payload if method == 'GET' else None

    # create request URL
    request_url = endpoint + canonical_uri

    # create and sign request
    creds = SimpleNamespace(
        access_key=access_key, secret_key=secret_key, token=session_token,
region=region,
    )

    request = AWSRequest(method=method, url=request_url, data=data, params=params)
    SigV4Auth(creds, service, region).add_auth(request)

    r = None

    # ************* SEND THE REQUEST *************
    if (method == 'GET'):

        print('++++ BEGIN GET REQUEST +++++')
        print('Request URL = ' + request_url)
        r = requests.get(request_url, headers=request.headers, verify=False,
params=params)

    elif (method == 'POST'):
```

```
        print('\n+++++ BEGIN POST REQUEST +++++')
        print('Request URL = ' + request_url)
        if (query_type == "loader"):
            request.headers['Content-type'] = 'application/json'
        r = requests.post(request_url, headers=request.headers, verify=False,
 data=data)

    else:
        print('Request method is neither "GET" nor "POST", something is wrong
 here.')

    if r is not None:
        print()
        print('+++++ RESPONSE +++++')
        print('Response code: %d\n' % r.status_code)
        response = r.text
        r.close()
        print(response)

        return response

help_msg = '''
    export AWS_ACCESS_KEY_ID=[MY_ACCESS_KEY_ID]
    export AWS_SECRET_ACCESS_KEY=[MY_SECRET_ACCESS_KEY]
    export AWS_SESSION_TOKEN=[MY_AWS_SESSION_TOKEN]
    export SERVICE_REGION=[us-east-1|us-east-2|us-west-2|eu-west-1]

    python version >=3.6 is required.

    Examples: For help
    python3 program_name.py -h

    Examples: Queries
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q status
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q sparql/
status
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q sparql -d
 "SELECT ?s WHERE { ?s ?p ?o }"
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a POST -q sparql -d
 "SELECT ?s WHERE { ?s ?p ?o }"
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a POST -q
 sparqlupdate -d "INSERT DATA { <https://s> <https://p> <https://o> }"
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q gremlin/
status
```

```
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q gremlin -d
 "g.V().count()"
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a POST -q gremlin -d
 "g.V().count()"
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q openCypher/
status
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q openCypher
 -d "MATCH (n1) RETURN n1 LIMIT 1;"
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a POST -q openCypher
 -d "MATCH (n1) RETURN n1 LIMIT 1;"
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q loader -d
 '{"loadId": "68b28dcc-8e15-02b1-133d-9bd0557607e6"}'
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q loader -d
 '{}'
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a POST -q loader
 -d '{"source": "source", "format" : "csv", "failOnError": "fail_on_error",
 "iamRoleArn": "iam_role_arn", "region": "region"}'

    Environment variables must be defined as AWS_ACCESS_KEY_ID,
 AWS_SECRET_ACCESS_KEY and SERVICE_REGION.
    You should also set AWS_SESSION_TOKEN environment variable if you are using
 temporary credentials (ex. IAM Role or EC2 Instance profile).

    Current Limitations:
    - Query mode "sparqlupdate" requires POST (as per the SPARQL 1.1 protocol)
            '''

def exit_and_print_help():
    print(help_msg)
    exit()

def parse_input_and_query_neptune():


    parser = ArgumentParser(description=help_msg,
 formatter_class=RawTextHelpFormatter)
    group_host = parser.add_mutually_exclusive_group()
    group_host.add_argument("-ho", "--host", type=str)
    group_port = parser.add_mutually_exclusive_group()
    group_port.add_argument("-p", "--port", type=int, help="port ex. 8182,
 default=8182", default=8182)
    group_action = parser.add_mutually_exclusive_group()
    group_action.add_argument("-a", "--action", type=str, help="http action,
 default = GET", default="GET")
```

```
    group_endpoint = parser.add_mutually_exclusive_group()
    group_endpoint.add_argument("-q", "--query_type", type=str, help="query_type,
 default = status ", default="status")
    group_data = parser.add_mutually_exclusive_group()
    group_data.add_argument("-d", "--data", type=str, help="data required for the
 http action", default="")

    args = parser.parse_args()
    print(args)

    # Read command line parameters
    host = args.host
    port = args.port
    method = args.action
    query_type = args.query_type
    query = args.data

    if (access_key == ''):
        print('!!! ERROR: Your AWS_ACCESS_KEY_ID environment variable is
 undefined.')
        exit_and_print_help()

    if (secret_key == ''):
        print('!!! ERROR: Your AWS_SECRET_ACCESS_KEY environment variable is
 undefined.')
        exit_and_print_help()

    if (region == ''):
        print('!!! ERROR: Your SERVICE_REGION environment variable is undefined.')
        exit_and_print_help()

    if host is None:
        print('!!! ERROR: Neptune DNS is missing')
        exit_and_print_help()

    host = host + ":" + str(port)
    make_signed_request(host, method, query_type, query)


if __name__ == "__main__":
    parse_input_and_query_neptune()
```

3. Em um terminal, navegue até o local do arquivo `neptunesigv4.py`.

4. Digite os comandos a seguir, substituindo a chave de acesso, a chave secreta e a região pelos valores corretos.

```
export AWS_ACCESS_KEY_ID=MY_ACCESS_KEY_ID
export AWS_SECRET_ACCESS_KEY=MY_SECRET_ACCESS_KEY
export SERVICE_REGION=us-east-1 or us-east-2 or us-west-1 or us-west-2 or ca-
central-1 or
                      sa-east-1 or eu-north-1 or eu-west-1 or eu-west-2 or eu-
west-3 or eu-central-1 or me-south-1 or
                      me-central-1 or il-central-1 or af-south-1 or ap-east-1 or
 ap-northeast-1 or ap-northeast-2 or ap-southeast-1 or ap-southeast-2 or ap-south-1
 or
                      cn-north-1 or cn-northwest-1 or
                      us-gov-east-1 or us-gov-west-1
```

   Se estiver usando credenciais temporárias, você deve especificar `AWS_SESSION_TOKEN` bem como `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, e `SERVICE_REGION`.

```
export AWS_SESSION_TOKEN=MY_AWS_SESSION_TOKEN
```

   > **Note**
   >
   > Se você estiver usando credenciais temporárias, elas expirarão após um intervalo especificado, incluindo o token de sessão.
   > Você deve atualizar o seu token de sessão ao solicitar novas credenciais. Para obter mais informações, consulte Using Temporary Security Credentials to Request Access to AWS Resources.

5. Digite um dos comandos a seguir para enviar uma solicitação assinada à instância de banco de dados do Neptune. Estes exemplos usam o Python versão 3.6.

   O status do endpoint

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q status
```

   Gremlin

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q gremlin -d
 "g.V().count()"
```

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a POST -q gremlin -d
 "g.V().count()"
```

Status do Gremlin

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q gremlin/
status
```

SPARQL

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q sparql -d
 "SELECT ?s WHERE { ?s ?p ?o }"
```

SPARQL UPDATE

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a POST -q sparqlupdate
 -d "INSERT DATA { <https://s> <https://p> <https://o> }"
```

Status do SPARQL

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q sparql/status
```

openCypher

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q openCypher -d
 "MATCH (n1) RETURN n1 LIMIT 1;"

python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a POST -q openCypher -
d "MATCH (n1) RETURN n1 LIMIT 1;"
```

Status do openCypher

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q openCypher/
status
```

Carregador

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q loader -d
 '{"loadId": "68b28dcc-8e15-02b1-133d-9bd0557607e6"}'

python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q loader -d
 '{}'

python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a POST -q loader
 -d '{"source": "source", "format" : "csv", "failOnError": "fail_on_error",
 "iamRoleArn": "iam_role_arn", "region": "region"}'
```

6. A sintaxe para executar o script do Python é a seguinte:

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p port -a GET|POST -q gremlin|
sparql|sparqlupdate|loader|status -d "string0data"
```

   O UPDATE SPARQL requer o POST.

Gerenciar o acesso usando políticas do IAM

As políticas do IAM são objetos JSON que definem permissões para usar ações e recursos.

Você controla o acesso AWS criando políticas e anexando-as a AWS identidades ou recursos. Uma política é um objeto AWS que, quando associada a uma identidade ou recurso, define suas permissões. AWS avalia essas políticas quando um principal (usuário, usuário raiz ou sessão de função) faz uma solicitação. As permissões nas políticas determinam se a solicitação será permitida ou negada. A maioria das políticas é armazenada AWS como documentos JSON. Para obter mais informações sobre a estrutura e o conteúdo de documentos de políticas JSON, consulte Visão geral das políticas JSON no Guia do Usuário do IAM.

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

Por padrão, usuários e funções não têm permissões. Para conceder aos usuários permissões para executar ações nos recursos que eles precisam, um administrador do IAM pode criar políticas do IAM. O administrador pode então adicionar as políticas do IAM aos perfis e os usuários podem presumir os perfis.

As políticas do IAM definem permissões para uma ação independente do método usado para executar a operação. Por exemplo, suponha que você tenha uma política que permite a ação

`iam:GetRole`. Um usuário com essa política pode obter informações de função da AWS Management Console AWS CLI, da ou da AWS API.

Políticas baseadas em identidade

As políticas baseadas em identidade são documentos de políticas de permissões JSON que você pode anexar a uma identidade, como usuário do IAM, grupo de usuários ou perfil. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Criando políticas do IAM no Guia do Usuário do IAM.

As políticas baseadas em identidade podem ser categorizadas ainda adicionalmente como políticas em linha ou políticas gerenciadas. As políticas em linha são anexadas diretamente a um único usuário, grupo ou perfil. As políticas gerenciadas são políticas autônomas que você pode associar a vários usuários, grupos e funções em seu Conta da AWS. As políticas AWS gerenciadas incluem políticas gerenciadas e políticas gerenciadas pelo cliente. Para saber como escolher entre uma política gerenciada ou uma política em linha, consulte Escolher entre políticas gerenciadas e políticas em linha no Guia do Usuário do IAM.

Usando políticas de controle de serviços (SCP) com organizações AWS

As políticas de controle de serviço (SCPs) são políticas JSON que especificam as permissões máximas para uma organização ou unidade organizacional (OU) em. AWS Organizations AWS Organizations é um serviço para agrupar e gerenciar centralmente várias AWS contas que sua empresa possui. Se você habilitar todos os atributos em uma organização, poderá aplicar políticas de controle de serviço (SCPs) a qualquer uma ou a todas as contas. O SCP limita as permissões para entidades nas contas dos membros, incluindo cada usuário raiz AWS da conta. Para obter mais informações sobre Organizations e SCPs, consulte Como os SCPs funcionam no Guia do AWS Organizations Usuário.

Clientes que implantam o Amazon Neptune AWS em uma conta AWS dentro de uma organização podem utilizar SCPs para controlar quais contas podem usar o Neptune. Para garantir o acesso ao Neptune em uma conta de membro, viabilize o acesso às ações do IAM do ambiente de gerenciamento e do plano de dados usando `neptune:*` e `neptune-db:*`, respectivamente.

Permissões necessárias para usar o console do Amazon Neptune

Para um usuário trabalhar com o console do Amazon Neptune, esse usuário deve ter um conjunto mínimo de permissões. Essas permissões possibilitam que o usuário descreva os recursos do

Neptune para a conta da AWS e forneça outras informações relacionadas, inclusive informações de segurança e rede do Amazon EC2.

Se você criar uma política do IAM que seja mais restritiva que as permissões mínimas necessárias, o console do não funcionará como pretendido para os usuários com essa política do IAM. Para garantir que esses usuários ainda consigam usar o console do Neptune, associe também a política gerenciada `NeptuneReadOnlyAccess` ao usuário, conforme descrito em AWS políticas gerenciadas (predefinidas) para o Amazon Neptune.

Você não precisa permitir permissões mínimas de console para usuários que estão fazendo chamadas somente para a API do Amazon Neptune AWS CLI ou para a Amazon Neptune.

Como associar uma política do IAM a um usuário do IAM

Para aplicar uma política gerenciada ou personalizada, associe-a a um usuário do IAM. Para obter um tutorial sobre esse tópico, consulte Criar e anexar sua primeira política gerenciada pelo cliente no Guia do usuário do IAM.

À medida que avança pelo tutorial, você pode usar um dos exemplos de política mostrados nessa seção como um ponto de partida e adequá-lo às suas necessidades. No fim do tutorial, você terá um usuário do IAM com uma política anexada que pode usar a ação `neptune-db:*`.

> **Important**
>
> - As alterações em uma política do IAM demoram até dez minutos para ser aplicadas aos recursos do Neptune especificados.
> - As políticas do IAM aplicadas a um cluster de banco de dados do Neptune aplicam-se a todas as instâncias desse cluster.

Usar diferentes tipos de política do IAM para controle de acesso ao Neptune

Para conceder acesso às ações administrativas do Neptune ou aos dados em um cluster de banco de dados do Neptune, associe políticas a um usuário ou um perfil do IAM. Para obter informações sobre como anexar uma política do IAM, consulte Como associar uma política do IAM a um usuário do IAM. Para obter informações sobre como associar uma política a um perfil, consulte Adding and Removing IAM Policies no Guia do usuário do IAM

Para ter acesso geral ao Neptune, é possível usar uma das políticas gerenciadas do Neptune. Para ter um acesso mais restrito, você pode criar a própria política personalizada usando as ações administrativas e os recursos compatíveis com o Neptune.

Em uma política personalizada do IAM, é possível usar dois tipos diferentes de declaração de política que controlam diferentes modos de acesso a um cluster de banco de dados do Neptune:

- Declarações de política administrativa: as declarações de política administrativa concedem acesso às APIs de gerenciamento do Neptune que você usa para criar, configurar e gerenciar um cluster de banco de dados e suas instâncias.

  Como o Neptune compartilha a funcionalidade com o Amazon RDS, as ações administrativas, os recursos e as chaves de condição nas políticas do Neptune usam um prefixo `rds:` por design.

- Declarações de política de acesso a dados: as declarações de política de acesso a dados usam ações de acesso a dados, recursos e chaves de condição para controlar o acesso aos dados contidos em um cluster de banco de dados.

  As ações de acesso a dados, os recursos e as chaves de condição do Neptune usam um prefixo `neptune-db:`.

Usar chaves de contexto de condição do IAM no Amazon Neptune

É possível especificar condições em uma declaração de política do IAM que controle o acesso ao Neptune. A declaração de política terá efeito apenas quando as condições forem verdadeiras.

Por exemplo, é recomendável que uma declaração de política só entre em vigor após uma data específica ou viabilize o acesso apenas quando um valor específico estiver presente na solicitação.

Para expressar condições, use chaves de condição predefinidas no elemento `Condition` de uma declaração de política com operadores de política de condição do IAM, como “igual a” ou “menos do que”.

Se você especificar vários elementos `Condition` em uma instrução ou várias chaves em um único `Condition` elemento, a AWS os avaliará usando uma operação lógica `AND`. Se você especificar vários valores para uma única chave de condição, AWS avalia a condição usando uma `OR` operação lógica. Todas as condições devem ser atendidas antes que as permissões da instrução sejam concedidas.

Você também pode usar variáveis de espaço reservado ao especificar condições. Por exemplo, é possível conceder a um usuário do IAM permissão para acessar um atributo somente se ele estiver marcado com seu nome de usuário do IAM. Para obter mais informações, consulte Elementos de política do IAM: variáveis e tags no Guia do usuário do IAM.

O tipo de dados de uma chave de condição determina quais operadores de condição você pode usar para comparar valores na solicitação com os valores na declaração de política. Se você usar um operador de condição que não seja compatível com esse tipo de dados, a correspondência sempre falhará, e a declaração da política nunca será aplicada.

O Neptune aceita conjuntos de chaves de condição para declarações de política administrativa diferentes dos aceitos para declarações de política de acesso a dados:

- Chaves de condição para declarações de política administrativa
- Chaves de condição para declarações de política de acesso a dados

Suporte para políticas do IAM e recursos de controle de acesso no Amazon Neptune

A tabela a seguir mostra quais atributos do IAM o Neptune aceita para declarações de política administrativa e declarações de política de acesso a dados:

Atributos do IAM que você pode usar com o Neptune

Atributo do IAM	Administração	Acesso aos dados
Políticas baseadas em identidade	Sim	Sim
Políticas baseadas em recursos	Não	Não
Ações das políticas	Sim	Sim
Atributos de políticas	Sim	Sim
Chaves de condições globais	Sim	(um subconjunto)

Atributo do IAM	Administração	Acesso aos dados
Chaves de condição baseadas em tags	Sim	Não
Listas de controle de acesso (ACLs)	Não	Não
Políticas de controle de serviço (SCPs)	Sim	Sim
Funções vinculadas ao serviço	Sim	Não

Limitações da política do IAM

As alterações em uma política do IAM demoram até dez minutos para ser aplicadas aos recursos do Neptune especificados.

As políticas do IAM aplicadas a um cluster de banco de dados do Neptune aplicam-se a todas as instâncias desse cluster.

No momento, o Neptune não é compatível com o controle de acesso entre contas.

AWS políticas gerenciadas (predefinidas) para o Amazon Neptune

AWS aborda muitos casos de uso comuns fornecendo políticas autônomas do IAM que são criadas e administradas pela AWS. As políticas gerenciadas concedem permissões necessárias para casos de uso comuns, de maneira que você possa evitar a necessidade de investigar quais permissões são necessárias. Para obter mais informações, consulte Políticas gerenciadas pela AWS no Guia do usuário do IAM.

As seguintes políticas AWS gerenciadas, que você pode anexar aos usuários em sua conta, são para usar as APIs de gerenciamento do Amazon Neptune:

- NeptuneReadOnlyAccess— Concede acesso somente de leitura a todos os recursos do Neptune para fins administrativos e de acesso a dados na conta raiz. AWS
- NeptuneFullAcesso — concede acesso total a todos os recursos do Neptune para fins administrativos e de acesso a dados na conta raiz. AWS Isso é recomendado se você precisar de

acesso total ao Neptune a partir AWS CLI do SDK ou, mas não para acesso. AWS Management Console

- NeptuneConsoleFullAccess— Concede acesso total na AWS conta raiz a todas as ações e recursos administrativos do Neptune, mas não a quaisquer ações ou recursos de acesso a dados. Também inclui permissões adicionais para simplificar o acesso ao Neptune pelo console, inclusive permissões limitadas do IAM e do Amazon EC2 (VPC).
- NeptuneGraphReadOnlyAccess — Fornece acesso somente de leitura a todos os recursos do Amazon Neptune Analytics, além de permissões somente de leitura para serviços dependentes
- AWSServiceRoleForNeptuneGraphPolicy— Permite que o Neptune Analytics crie gráficos para CloudWatch publicar métricas e registros operacionais e de uso.

As políticas e os perfis do IAM no Neptune concedem algum acesso aos recursos do Amazon RDS, pois o Neptune compartilha tecnologia operacional com o Amazon RDS para determinados atributos de gerenciamento. Isso inclui permissões administrativas da API, e é por isso que as ações administrativas do Neptune têm um prefixo `rds:`.

Atualizações nas políticas gerenciadas do AWS Neptune

A seguinte tabela monitora as atualizações das políticas gerenciadas do Neptune a partir do momento em que o Neptune começou a monitorar essas alterações:

Política	Descrição	Data
AWS políticas gerenciadas para o Amazon Neptune — atualização das políticas existentes	As políticas `NeptuneReadOnlyAcess` `NeptuneFullAccess` gerenciadas agora incluem `Sid` (ID da declaração) como um identificador na declaração de política.	2024-01-22
NeptuneGraphReadOnlyAccess (lançado)	Lançado para fornecer acesso somente leitura aos grafos e recursos do Neptune Analytics .	2023-11-29
AWSServiceRoleForNeptuneGraphPolicy(lançado)	Lançado para permitir o acesso aos gráficos do	2023-11-29

Política	Descrição	Data
	Neptune Analytics CloudWatch para publicar métricas e registros operacionais e de uso. Consulte Using service-linked roles (SLRs) in Neptune Analytics.	
NeptuneConsoleFullAccess(permissões adicionadas)	As permissões adicionadas dão todo o acesso necessário para interagir com os grafos do Neptune Analytics.	2023-11/29
NeptuneFullAcesso (permissões adicionadas)	Foram adicionadas permissões de acesso a dados e permissões para novas APIs de banco de dados globais.	2022-07-28
NeptuneConsoleFullAccess(permissões adicionadas)	Foram adicionadas permissões para novas APIs de banco de dados globais.	2022-07-21
O Neptune iniciou o rastreamento das alterações	A Neptune começou a monitorar as mudanças em suas políticas gerenciadas AWS .	2022-07-21

Política do AWS gerenciada pela **`NeptuneReadOnlyAccess`**

A política NeptuneReadOnlyAccessgerenciada abaixo concede acesso somente de leitura a todas as ações e recursos do Neptune para fins administrativos e de acesso a dados.

> **Note**
>
> Essa política foi atualizada em 21/07/2022 para incluir permissões de acesso a dados somente leitura, bem como permissões administrativas somente leitura, além de incluir permissões para ações globais do banco de dados.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowReadOnlyPermissionsForRDS",
            "Effect": "Allow",
            "Action": [
                "rds:DescribeAccountAttributes",
                "rds:DescribeCertificates",
                "rds:DescribeDBClusterParameterGroups",
                "rds:DescribeDBClusterParameters",
                "rds:DescribeDBClusterSnapshotAttributes",
                "rds:DescribeDBClusterSnapshots",
                "rds:DescribeDBClusters",
                "rds:DescribeDBEngineVersions",
                "rds:DescribeDBInstances",
                "rds:DescribeDBLogFiles",
                "rds:DescribeDBParameterGroups",
                "rds:DescribeDBParameters",
                "rds:DescribeDBSubnetGroups",
                "rds:DescribeEventCategories",
                "rds:DescribeEventSubscriptions",
                "rds:DescribeEvents",
                "rds:DescribeGlobalClusters",
                "rds:DescribeOrderableDBInstanceOptions",
                "rds:DescribePendingMaintenanceActions",
                "rds:DownloadDBLogFilePortion",
                "rds:ListTagsForResource"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowReadOnlyPermissionsForCloudwatch",
            "Effect": "Allow",
            "Action": [
```

```
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowReadOnlyPermissionsForEC2",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeInternetGateways",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeVpcs"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowReadOnlyPermissionsForKMS",
            "Effect": "Allow",
            "Action": [
                "kms:ListKeys",
                "kms:ListRetirableGrants",
                "kms:ListAliases",
                "kms:ListKeyPolicies"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowReadOnlyPermissionsForLogs",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams",
                "logs:GetLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:*:*:log-group:/aws/rds/*:log-stream:*",
                "arn:aws:logs:*:*:log-group:/aws/neptune/*:log-stream:*"
            ]
        },
        {
            "Sid": "AllowReadOnlyPermissionsForNeptuneDB",
```

```
            "Effect": "Allow",
            "Action": [
                "neptune-db:Read*",
                "neptune-db:Get*",
                "neptune-db:List*"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Política do AWS gerenciada pela **NeptuneFullAccess**

A política de NeptuneFullacesso gerenciado abaixo concede acesso total a todas as ações e recursos do Neptune para fins administrativos e de acesso a dados. É recomendável se você precisar de acesso total do AWS CLI ou de um SDK, mas não do AWS Management Console.

> **Note**
>
> Essa política foi atualizada em 21/07/2022 para incluir permissões de acesso a dados total, bem como permissões administrativas totais, além de incluir permissões para ações globais do banco de dados.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowNeptuneCreate",
            "Effect": "Allow",
            "Action": [
                "rds:CreateDBCluster",
                "rds:CreateDBInstance"
            ],
            "Resource": [
                "arn:aws:rds:*:*:*"
            ],
            "Condition": {
                "StringEquals": {
                    "rds:DatabaseEngine": [
```

```
                    "graphdb",
                    "neptune"
                ]
            }
        }
    },
    {
        "Sid": "AllowManagementPermissionsForRDS",
        "Effect": "Allow",
        "Action": [
            "rds:AddRoleToDBCluster",
            "rds:AddSourceIdentifierToSubscription",
            "rds:AddTagsToResource",
            "rds:ApplyPendingMaintenanceAction",
            "rds:CopyDBClusterParameterGroup",
            "rds:CopyDBClusterSnapshot",
            "rds:CopyDBParameterGroup",
            "rds:CreateDBClusterEndpoint",
            "rds:CreateDBClusterParameterGroup",
            "rds:CreateDBClusterSnapshot",
            "rds:CreateDBParameterGroup",
            "rds:CreateDBSubnetGroup",
            "rds:CreateEventSubscription",
            "rds:CreateGlobalCluster",
            "rds:DeleteDBCluster",
            "rds:DeleteDBClusterEndpoint",
            "rds:DeleteDBClusterParameterGroup",
            "rds:DeleteDBClusterSnapshot",
            "rds:DeleteDBInstance",
            "rds:DeleteDBParameterGroup",
            "rds:DeleteDBSubnetGroup",
            "rds:DeleteEventSubscription",
            "rds:DeleteGlobalCluster",
            "rds:DescribeDBClusterEndpoints",
            "rds:DescribeAccountAttributes",
            "rds:DescribeCertificates",
            "rds:DescribeDBClusterParameterGroups",
            "rds:DescribeDBClusterParameters",
            "rds:DescribeDBClusterSnapshotAttributes",
            "rds:DescribeDBClusterSnapshots",
            "rds:DescribeDBClusters",
            "rds:DescribeDBEngineVersions",
            "rds:DescribeDBInstances",
            "rds:DescribeDBLogFiles",
```

```
                "rds:DescribeDBParameterGroups",
                "rds:DescribeDBParameters",
                "rds:DescribeDBSecurityGroups",
                "rds:DescribeDBSubnetGroups",
                "rds:DescribeEngineDefaultClusterParameters",
                "rds:DescribeEngineDefaultParameters",
                "rds:DescribeEventCategories",
                "rds:DescribeEventSubscriptions",
                "rds:DescribeEvents",
                "rds:DescribeGlobalClusters",
                "rds:DescribeOptionGroups",
                "rds:DescribeOrderableDBInstanceOptions",
                "rds:DescribePendingMaintenanceActions",
                "rds:DescribeValidDBInstanceModifications",
                "rds:DownloadDBLogFilePortion",
                "rds:FailoverDBCluster",
                "rds:FailoverGlobalCluster",
                "rds:ListTagsForResource",
                "rds:ModifyDBCluster",
                "rds:ModifyDBClusterEndpoint",
                "rds:ModifyDBClusterParameterGroup",
                "rds:ModifyDBClusterSnapshotAttribute",
                "rds:ModifyDBInstance",
                "rds:ModifyDBParameterGroup",
                "rds:ModifyDBSubnetGroup",
                "rds:ModifyEventSubscription",
                "rds:ModifyGlobalCluster",
                "rds:PromoteReadReplicaDBCluster",
                "rds:RebootDBInstance",
                "rds:RemoveFromGlobalCluster",
                "rds:RemoveRoleFromDBCluster",
                "rds:RemoveSourceIdentifierFromSubscription",
                "rds:RemoveTagsFromResource",
                "rds:ResetDBClusterParameterGroup",
                "rds:ResetDBParameterGroup",
                "rds:RestoreDBClusterFromSnapshot",
                "rds:RestoreDBClusterToPointInTime",
                "rds:StartDBCluster",
                "rds:StopDBCluster"
            ],
            "Resource": [
                "*"
            ]
        },
```

```
        {
            "Sid": "AllowOtherDepedentPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics",
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeVpcs",
                "kms:ListAliases",
                "kms:ListKeyPolicies",
                "kms:ListKeys",
                "kms:ListRetirableGrants",
                "logs:DescribeLogStreams",
                "logs:GetLogEvents",
                "sns:ListSubscriptions",
                "sns:ListTopics",
                "sns:Publish"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "AllowPassRoleForNeptune",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:passedToService": "rds.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AllowCreateSLRForNeptune",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
            "Condition": {
```

```
                "StringLike": {
                    "iam:AWSServiceName": "rds.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AllowDataAccessForNeptune",
            "Effect": "Allow",
            "Action": [
                "neptune-db:*"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Política do AWS gerenciada pela **NeptuneConsoleFullAccess**

A política NeptuneConsoleFullAccessgerenciada abaixo concede acesso total a todas as ações e recursos do Neptune para fins administrativos, mas não para fins de acesso a dados. Também inclui permissões adicionais para simplificar o acesso ao Neptune pelo console, inclusive permissões limitadas do IAM e do Amazon EC2 (VPC).

> **Note**
>
> Esta política foi atualizada em 29/11/2023 para incluir as permissões necessárias para interagir com os grafos do Neptune Analytics.
> Atualizada em 21/07/2022 para incluir permissões para ações globais do banco de dados.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowNeptuneCreate",
      "Effect": "Allow",
      "Action": [
        "rds:CreateDBCluster",
        "rds:CreateDBInstance"
```

```
      ],
      "Resource": [
        "arn:aws:rds:*:*:*"
      ],
      "Condition": {
        "StringEquals": {
          "rds:DatabaseEngine": [
            "graphdb",
            "neptune"
          ]
        }
      }
    },
    {
      "Sid": "AllowManagementPermissionsForRDS",
      "Action": [
        "rds:AddRoleToDBCluster",
        "rds:AddSourceIdentifierToSubscription",
        "rds:AddTagsToResource",
        "rds:ApplyPendingMaintenanceAction",
        "rds:CopyDBClusterParameterGroup",
        "rds:CopyDBClusterSnapshot",
        "rds:CopyDBParameterGroup",
        "rds:CreateDBClusterParameterGroup",
        "rds:CreateDBClusterSnapshot",
        "rds:CreateDBParameterGroup",
        "rds:CreateDBSubnetGroup",
        "rds:CreateEventSubscription",
        "rds:DeleteDBCluster",
        "rds:DeleteDBClusterParameterGroup",
        "rds:DeleteDBClusterSnapshot",
        "rds:DeleteDBInstance",
        "rds:DeleteDBParameterGroup",
        "rds:DeleteDBSubnetGroup",
        "rds:DeleteEventSubscription",
        "rds:DescribeAccountAttributes",
        "rds:DescribeCertificates",
        "rds:DescribeDBClusterParameterGroups",
        "rds:DescribeDBClusterParameters",
        "rds:DescribeDBClusterSnapshotAttributes",
        "rds:DescribeDBClusterSnapshots",
        "rds:DescribeDBClusters",
        "rds:DescribeDBEngineVersions",
        "rds:DescribeDBInstances",
```

```
        "rds:DescribeDBLogFiles",
        "rds:DescribeDBParameterGroups",
        "rds:DescribeDBParameters",
        "rds:DescribeDBSecurityGroups",
        "rds:DescribeDBSubnetGroups",
        "rds:DescribeEngineDefaultClusterParameters",
        "rds:DescribeEngineDefaultParameters",
        "rds:DescribeEventCategories",
        "rds:DescribeEventSubscriptions",
        "rds:DescribeEvents",
        "rds:DescribeOptionGroups",
        "rds:DescribeOrderableDBInstanceOptions",
        "rds:DescribePendingMaintenanceActions",
        "rds:DescribeValidDBInstanceModifications",
        "rds:DownloadDBLogFilePortion",
        "rds:FailoverDBCluster",
        "rds:ListTagsForResource",
        "rds:ModifyDBCluster",
        "rds:ModifyDBClusterParameterGroup",
        "rds:ModifyDBClusterSnapshotAttribute",
        "rds:ModifyDBInstance",
        "rds:ModifyDBParameterGroup",
        "rds:ModifyDBSubnetGroup",
        "rds:ModifyEventSubscription",
        "rds:PromoteReadReplicaDBCluster",
        "rds:RebootDBInstance",
        "rds:RemoveRoleFromDBCluster",
        "rds:RemoveSourceIdentifierFromSubscription",
        "rds:RemoveTagsFromResource",
        "rds:ResetDBClusterParameterGroup",
        "rds:ResetDBParameterGroup",
        "rds:RestoreDBClusterFromSnapshot",
        "rds:RestoreDBClusterToPointInTime"
      ],
      "Effect": "Allow",
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "AllowOtherDepedentPermissions",
      "Action": [
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:ListMetrics",
```

```
        "ec2:AllocateAddress",
        "ec2:AssignIpv6Addresses",
        "ec2:AssignPrivateIpAddresses",
        "ec2:AssociateAddress",
        "ec2:AssociateRouteTable",
        "ec2:AssociateSubnetCidrBlock",
        "ec2:AssociateVpcCidrBlock",
        "ec2:AttachInternetGateway",
        "ec2:AttachNetworkInterface",
        "ec2:CreateCustomerGateway",
        "ec2:CreateDefaultSubnet",
        "ec2:CreateDefaultVpc",
        "ec2:CreateInternetGateway",
        "ec2:CreateNatGateway",
        "ec2:CreateNetworkInterface",
        "ec2:CreateRoute",
        "ec2:CreateRouteTable",
        "ec2:CreateSecurityGroup",
        "ec2:CreateSubnet",
        "ec2:CreateVpc",
        "ec2:CreateVpcEndpoint",
        "ec2:CreateVpcEndpoint",
        "ec2:DescribeAccountAttributes",
        "ec2:DescribeAccountAttributes",
        "ec2:DescribeAddresses",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeCustomerGateways",
        "ec2:DescribeInstances",
        "ec2:DescribeNatGateways",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribePrefixLists",
        "ec2:DescribeRouteTables",
        "ec2:DescribeSecurityGroupReferences",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcAttribute",
        "ec2:DescribeVpcAttribute",
        "ec2:DescribeVpcEndpoints",
        "ec2:DescribeVpcs",
        "ec2:DescribeVpcs",
        "ec2:ModifyNetworkInterfaceAttribute",
```

```
        "ec2:ModifySubnetAttribute",
        "ec2:ModifyVpcAttribute",
        "ec2:ModifyVpcEndpoint",
        "iam:ListRoles",
        "kms:ListAliases",
        "kms:ListKeyPolicies",
        "kms:ListKeys",
        "kms:ListRetirableGrants",
        "logs:DescribeLogStreams",
        "logs:GetLogEvents",
        "sns:ListSubscriptions",
        "sns:ListTopics",
        "sns:Publish"
      ],
      "Effect": "Allow",
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "AllowPassRoleForNeptune",
      "Action": "iam:PassRole",
      "Effect": "Allow",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:passedToService": "rds.amazonaws.com"
        }
      }
    },
    {
      "Sid": "AllowCreateSLRForNeptune",
      "Action": "iam:CreateServiceLinkedRole",
      "Effect": "Allow",
      "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
      "Condition": {
        "StringLike": {
          "iam:AWSServiceName": "rds.amazonaws.com"
        }
      }
    },
    {
      "Sid": "AllowManagementPermissionsForNeptuneAnalytics",
```

```
      "Effect": "Allow",
      "Action": [
        "neptune-graph:CreateGraph",
        "neptune-graph:DeleteGraph",
        "neptune-graph:GetGraph",
        "neptune-graph:ListGraphs",
        "neptune-graph:UpdateGraph",
        "neptune-graph:ResetGraph",
        "neptune-graph:CreateGraphSnapshot",
        "neptune-graph:DeleteGraphSnapshot",
        "neptune-graph:GetGraphSnapshot",
        "neptune-graph:ListGraphSnapshots",
        "neptune-graph:RestoreGraphFromSnapshot",
        "neptune-graph:CreatePrivateGraphEndpoint",
        "neptune-graph:GetPrivateGraphEndpoint",
        "neptune-graph:ListPrivateGraphEndpoints",
        "neptune-graph:DeletePrivateGraphEndpoint",
        "neptune-graph:CreateGraphUsingImportTask",
        "neptune-graph:GetImportTask",
        "neptune-graph:ListImportTasks",
        "neptune-graph:CancelImportTask"
      ],
      "Resource": [
        "arn:aws:neptune-graph:*:*:*"
      ]
    },
    {
      "Sid": "AllowPassRoleForNeptuneAnalytics",
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:passedToService": "neptune-graph.amazonaws.com"
        }
      }
    },
    {
      "Sid": "AllowCreateSLRForNeptuneAnalytics",
      "Effect": "Allow",
      "Action": "iam:CreateServiceLinkedRole",
      "Resource": "arn:aws:iam::*:role/aws-service-role/neptune-graph.amazonaws.com/
AWSServiceRoleForNeptuneGraph",
      "Condition": {
```

```
        "StringLike": {
          "iam:AWSServiceName": "neptune-graph.amazonaws.com"
        }
      }
    }
  ]
}
```

Política do AWS gerenciada pela **NeptuneGraphReadOnlyAccess**

A política de NeptuneGraphReadOnlyacesso gerenciado abaixo fornece acesso somente de leitura a todos os recursos do Amazon Neptune Analytics, além de permissões somente de leitura para serviços dependentes.

Esta política inclui permissões para fazer o seguinte:

- No Amazon EC2: recupere informações sobre VPCs, sub-redes, grupos de segurança e zonas de disponibilidade.
- Para AWS KMS — Recupere informações sobre chaves e aliases do KMS.
- Para CloudWatch — Recupere informações sobre CloudWatch métricas.
- Para CloudWatch registros — recupere informações sobre fluxos de CloudWatch registros e eventos.

> **Note**
>
> Esta política foi lançada em 29/11/2023.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowReadOnlyPermissionsForNeptuneGraph",
      "Effect": "Allow",
      "Action": [
        "neptune-graph:Get*",
        "neptune-graph:List*",
        "neptune-graph:Read*"
      ],
```

```
      "Resource": "*"
    },
    {
      "Sid": "AllowReadOnlyPermissionsForEC2",
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeVpcEndpoints",
        "ec2:DescribeVpcAttribute",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcs",
        "ec2:DescribeAvailabilityZones"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowReadOnlyPermissionsForKMS",
      "Effect": "Allow",
      "Action": [
        "kms:ListKeys",
        "kms:ListAliases"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowReadOnlyPermissionsForCloudwatch",
      "Effect": "Allow",
      "Action": [
        "cloudwatch:GetMetricData",
        "cloudwatch:ListMetrics",
        "cloudwatch:GetMetricStatistics"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowReadOnlyPermissionsForLogs",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogStreams",
        "logs:GetLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:/aws/neptune/*:log-stream:*"
      ]
```

```
    }
  ]
}
```

Política do AWS gerenciada pela **AWSServiceRoleForNeptuneGraphPolicy**

A política AWSServiceRoleForNeptuneGraphPolicygerenciada abaixo fornece acesso a gráficos CloudWatch para publicar métricas e registros operacionais e de uso. Consulte nan-service-linked-roles.

> **Note**
>
> Esta política foi lançada em 29/11/2023.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "GraphMetrics",
      "Effect": "Allow",
      "Action": [
        "cloudwatch:PutMetricData"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "cloudwatch:namespace": [
            "AWS/Neptune",
            "AWS/Usage"
          ]
        }
      }
    },
    {
      "Sid": "GraphLogGroup",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:/aws/neptune/*"
```

```
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "${aws:PrincipalAccount}"
        }
      }
    },
    {
      "Sid": "GraphLogEvents",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:/aws/neptune/*:log-stream:*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "${aws:PrincipalAccount}"
        }
      }
    }
  ]
}
```

Chaves de contexto de condição do IAM compatíveis com o Amazon Neptune

É possível especificar as condições nas políticas do IAM que controlam às ações de gerenciamento e aos recursos do Neptune. A declaração de política terá efeito apenas quando as condições forem verdadeiras.

Por exemplo, é recomendável que uma declaração de política só entre em vigor após uma data específica ou viabilize o acesso apenas quando um valor específico estiver presente na solicitação da API.

Para expressar condições, use chaves de condição predefinidas no elemento `Condition` de uma declaração de política com operadores de política de condição do IAM, como “igual a” ou “menos do que”.

Se você especificar vários elementos `Condition` em uma instrução ou várias chaves em um único `Condition` elemento, a AWS os avaliará usando uma operação lógica `AND`. Se você especificar vários valores para uma única chave de condição, AWS avalia a condição usando uma `OR` operação lógica. Todas as condições devem ser atendidas antes que as permissões da instrução sejam concedidas.

Você também pode usar variáveis de espaço reservado ao especificar condições. Por exemplo, é possível conceder a um usuário do IAM permissão para acessar um atributo somente se ele estiver marcado com seu nome de usuário do IAM. Para obter mais informações, consulte Elementos de política do IAM: variáveis e tags no Guia do usuário do IAM.

O tipo de dados de uma chave de condição determina quais operadores de condição você pode usar para comparar valores na solicitação com os valores na declaração de política. Se você usar um operador de condição que não seja compatível com esse tipo de dados, a correspondência sempre falhará, e a declaração da política nunca será aplicada.

Chaves de condição do IAM para declarações de política administrativa do Neptune

- Chaves de condição globais — Você pode usar a maioria das chaves de condição AWS globais nas declarações de política administrativa do Neptune.
- Chaves de condição específicas do serviço — são chaves definidas para serviços específicos AWS . Aquelas compatíveis com o Neptune para declarações de política administrativa estão listadas em Chaves de condição disponíveis em declarações de política administrativa do IAM do Neptune.

Chaves de condição do IAM para declarações de política de acesso a dados do Neptune

- Chaves de condição globais: o subconjunto dessas chaves aceitas pelo Neptune nas declarações de política de acesso a dados está listado em AWS chaves de contexto de condição global suportadas pelo Neptune em declarações de política de acesso a dados.
- As chaves de condição específicas do serviço que o Neptune define para declarações de política de acesso a dados estão listadas em Chaves de condição.

Declarações personalizadas de política administrativa do IAM para o Amazon Neptune

As declarações de política administrativa permitem controlar o que um usuário do IAM pode fazer para gerenciar um banco de dados Neptune.

Uma declaração de política administrativa do Neptune concede acesso a uma ou ações administrativas e recursos administrativos aceitos pelo Neptune. Você também pode usar Chaves de condição para tornar as permissões administrativas mais específicas.

> **Note**
>
> Como o Neptune compartilha a funcionalidade com o Amazon RDS, as ações administrativas, os recursos e as chaves de condição específicas do serviço em declarações de política administrativa usam um prefixo `rds:` por design.

Tópicos

Ações disponíveis nas declarações de política administrativa do IAM do Neptune

É possível usar as ações administrativas listadas abaixo no elemento `Action` de uma declaração de política do IAM para controlar o acesso às APIs de gerenciamento do Neptune. Quando usa uma ação em uma política, você geralmente permite ou nega acesso à operação da API ou ao comando da CLI com o mesmo nome. No entanto, em alguns casos, uma única ação controla o acesso a mais de uma operação. Como alternativa, algumas operações exigem várias ações diferentes.

O campo `Resource type` na lista abaixo indica se cada ação é compatível com permissões em nível de recurso. Se não houver valor para esse campo, você deverá especificar todos os recursos ("*") no elemento `Resource` de sua declaração de política. Se a coluna incluir um tipo de recurso, você poderá especificar um ARN de recurso desse tipo em uma declaração com essa ação. Os tipos de recurso administrativo do Neptune estão listados nesta página.

Os recursos obrigatórios são indicados na lista abaixo com um asterisco (*). Se você especificar um ARN de permissão no nível do recurso em uma instrução que esteja usando essa ação, ele deverá ser desse tipo. Algumas ações oferecem suporte a vários tipos de recursos. Se um tipo de recurso for opcional (em outras palavras, não estiver marcado com um asterisco), não será necessário incluí-lo.

Para obter mais informações sobre os campos listados aqui, consulte action table no Guia do usuário do IAM.

rds: ToDBCluster AddRole

`AddRoleToDBCluster` associa um perfil do IAM a um cluster de banco de dados do Neptune.

Nível de acesso: `Write`.

Ações dependentes: `iam:PassRole`.

Tipo de recurso: cluster (obrigatório).

rds: Assinatura AddSource IdentifierTo

`AddSourceIdentifierToSubscription` adiciona um identificador de origem a uma assinatura de notificações de eventos existente do Neptune.

Nível de acesso: `Write`.

Tipo de recurso: es (obrigatório).

vermelhos: AddTags ToResource

`AddTagsToResource` associa um perfil do IAM a um cluster de banco de dados do Neptune.

Nível de acesso: `Write`.

Tipos de recurso:

- db
- es
- pg
- cluster-snapshot
- subgrp

Chaves de condição:

- aws:RequestTag//tecla de *tag*
- leis: TagKeys

vermelhos: ApplyPending MaintenanceAction

`ApplyPendingMaintenanceAction` aplica uma ação de manutenção pendente a um recurso.

Nível de acesso: `Write`.

Tipo de recurso: db (obrigatório).

Grupo RDS: CopyDB ClusterParameter

`CopyDBClusterParameterGroup` copia o grupo de parâmetros de cluster de banco de dados especificado.

Nível de acesso: `Write`.

Tipo de recurso: cluster-pg (obrigatório).

RDS: Copiar banco de dados ClusterSnapshot

`CopyDBClusterSnapshot` copia um snapshot de um cluster de banco de dados.

Nível de acesso: `Write`.

Tipo de recurso: cluster-snapshot (obrigatório).

RDS: Copiar banco de dados ParameterGroup

`CopyDBParameterGroup` copia o grupo de parâmetros de banco de dados especificado.

Nível de acesso: `Write`.

Tipo de recurso: pg (obrigatório).

rds:CreateDBCluster

`CreateDBCluster` cria um cluster de banco de dados do Neptune.

Nível de acesso: `Tagging`.

Ações dependentes: `iam:PassRole`.

Tipos de recurso:

- cluster (obrigatório).
- cluster-pg (obrigatório).
- subgrp (obrigatório).

Chaves de condição:

- aws:RequestTag//tecla de *tag*
- leis: TagKeys
- netuno-rds_ DatabaseEngine

RDS: Grupo DB criado ClusterParameter

`CreateDBClusterParameterGroup` cria um grupo de parâmetros de cluster de banco de dados.

Nível de acesso: `Tagging`.

Tipo de recurso: cluster-pg (obrigatório).

Chaves de condição:

- aws:RequestTag//tecla de *tag*
- leis: TagKeys

RDS: criado por B ClusterSnapshot

`CreateDBClusterSnapshot` cria um snapshot de um cluster de banco de dados.

Nível de acesso: `Tagging`.

Tipos de recurso:

- cluster (obrigatório).
- cluster-snapshot (obrigatório).

Chaves de condição:

- aws:RequestTag//tecla de *tag*
- leis: TagKeys

rds:CreateDBInstance

`CreateDBInstance` cria uma instância de banco de dados.

Nível de acesso: `Tagging`.

Ações dependentes: `iam:PassRole`.

Tipos de recurso:

- db (obrigatório).
- pg (obrigatório).
- subgrp (obrigatório).

Chaves de condição:

- aws:RequestTag//tecla de *tag*
- leis: TagKeys

RDS: criado por B ParameterGroup

`CreateDBParameterGroup` cria um novo grupo de parâmetros de banco de dados.

Nível de acesso: `Tagging`.

Tipo de recurso: pg (obrigatório).

Chaves de condição:

- aws:RequestTag//tecla de *tag*
- leis: TagKeys

RDS: criado por B SubnetGroup

`CreateDBSubnetGroup` cria um grupo de sub-redes de banco de dados.

Nível de acesso: `Tagging`.

Tipo de recurso: subgrp (obrigatório).

Chaves de condição:

- aws:RequestTag//tecla de *tag*
- leis: TagKeys

rds: Assinatura CreateEvent

`CreateEventSubscription` cria uma assinatura de notificação de eventos do Neptune.

Nível de acesso: `Tagging`.

Tipo de recurso: es (obrigatório).

Chaves de condição:

- aws:RequestTag//tecla de *tag*
- leis: TagKeys

rds:DeleteDBCluster

`DeleteDBCluster` exclui um cluster de banco de dados do Neptune existente.

Nível de acesso: `Write`.

Tipos de recurso:

- cluster (obrigatório).
- cluster-snapshot (obrigatório).

RDS: Excluir grupo DB ClusterParameter

`DeleteDBClusterParameterGroup` exclui um grupo de parâmetros de cluster de banco de dados especificado.

Nível de acesso: `Write`.

Tipo de recurso: cluster-pg (obrigatório).

RDS: Excluir banco de dados ClusterSnapshot

DeleteDBClusterSnapshot exclui um snapshot do cluster de banco de dados.

Nível de acesso: Write.

Tipo de recurso: cluster-snapshot (obrigatório).

rds:DeleteDBInstance

DeleteDBInstance exclui uma instância de banco de dados especificada.

Nível de acesso: Write.

Tipo de recurso: db (obrigatório).

RDS: Excluir banco de dados ParameterGroup

DeleteDBParameterGroupexclui um banco de dados ParameterGroup especificado.

Nível de acesso: Write.

Tipo de recurso: pg (obrigatório).

RDS: Excluir banco de dados SubnetGroup

DeleteDBSubnetGroup exclui um grupo de sub-redes de banco de dados.

Nível de acesso: Write.

Tipo de recurso: subgrp (obrigatório).

rds: Assinatura DeleteEvent

DeleteEventSubscription exclui uma assinatura de notificações de eventos.

Nível de acesso: Write.

Tipo de recurso: es (obrigatório).

RDS: grupos de banco ClusterParameter de dados descritos

DescribeDBClusterParameterGroupsretorna uma lista de ClusterParameterGroup descrições do banco de dados.

Nível de acesso: List.

Tipo de recurso: cluster-pg (obrigatório).

RDS: DB descrito ClusterParameters

DescribeDBClusterParameters gera a lista de parâmetros detalhada de um grupo de parâmetros de cluster de banco de dados.

Nível de acesso: List.

Tipo de recurso: cluster-pg (obrigatório).

RDS: atributos de banco ClusterSnapshot de dados descritos

DescribeDBClusterSnapshotAttributes gera uma lista de nomes e valores de atributos de snapshot do cluster de banco de dados de um snapshot manual do cluster de banco de dados.

Nível de acesso: List.

Tipo de recurso: cluster-snapshot (obrigatório).

RDS: DB descrito ClusterSnapshots

DescribeDBClusterSnapshots gera informações sobre snapshots do cluster banco de dados.

Nível de acesso: Read.

rds:DescribeDBClusters

DescribeDBClusters gera informações sobre um cluster de banco de dados provisionados do Neptune.

Nível de acesso: List.

Tipo de recurso: cluster (obrigatório).

RDS: DB descrito EngineVersions

DescribeDBEngineVersions gera uma lista dos mecanismos de banco de dados disponíveis.

Nível de acesso: List.

Tipo de recurso: pg (obrigatório).

rds:DescribeDBInstances

DescribeDBInstances gera informações sobre instâncias de banco de dados.

Nível de acesso: `List`.

Tipo de recurso: es (obrigatório).

RDS: DB descrito ParameterGroups

`DescribeDBParameterGroups`retorna uma lista de ParameterGroup descrições do banco de dados.

Nível de acesso: `List`.

Tipo de recurso: pg (obrigatório).

rds:DescribeDBParameters

`DescribeDBParameters` exibe a lista de parâmetros detalhada de um grupo de parâmetros de banco de dados específico.

Nível de acesso: `List`.

Tipo de recurso: pg (obrigatório).

RDS: DB descrito SubnetGroups

`DescribeDBSubnetGroups`retorna uma lista de SubnetGroup descrições do banco de dados.

Nível de acesso: `List`.

Tipo de recurso: subgrp (obrigatório).

rds: Categorias DescribeEvent

`DescribeEventCategories` exibe uma lista de categorias de todos os tipos de origem de eventos ou, se especificado, de um determinado tipo de origem.

Nível de acesso: `List`.

rds: Assinaturas DescribeEvent

`DescribeEventSubscriptions` lista todas as descrições de assinaturas de uma conta de cliente.

Nível de acesso: `List`.

Tipo de recurso: es (obrigatório).

vermelhos: DescribeEvents

`DescribeEvents` exibe os eventos relacionados a instâncias de bancos de dados, grupos de segurança de banco de dados e grupos de parâmetros de banco de dados dos últimos 14 dias.

Nível de acesso: `List`.

Tipo de recurso: es (obrigatório).

Vermelho: DB DescribeOrderable InstanceOptions

`DescribeOrderableDBInstanceOptions` gera uma lista de opções de instância de banco de dados que podem ser solicitadas para o mecanismo especificado.

Nível de acesso: `List`.

vermelhos: DescribePending MaintenanceActions

`DescribePendingMaintenanceActions` gera uma lista de recursos (por exemplo, instâncias de banco de dados) que têm pelo menos uma ação de manutenção pendente.

Nível de acesso: `List`.

Tipo de recurso: db (obrigatório).

Vermelho: DB DescribeValid InstanceModifications

`DescribeValidDBInstanceModifications` lista as modificações disponíveis que podem ser realizadas na instância de banco de dados.

Nível de acesso: `List`.

Tipo de recurso: db (obrigatório).

rds:FailoverDBCluster

`FailoverDBCluster` força um failover para um cluster de banco de dados.

Nível de acesso: `Write`.

Tipo de recurso: cluster (obrigatório).

vermelhos: ListTags ForResource

`ListTagsForResource` lista todas as tags em um recurso do Neptune.

Nível de acesso: `Read`.

Tipos de recurso:

- cluster-snapshot
- db
- es
- pg
- subgrp

rds:ModifyDBCluster

`ModifyDBCluster`

Modifica uma configuração de um cluster de banco de dados do Neptune.

Nível de acesso: `Write`.

Ações dependentes: `iam:PassRole`.

Tipos de recurso:

- cluster (obrigatório).
- cluster-pg (obrigatório).

RDS: ModifyDB Group ClusterParameter

`ModifyDBClusterParameterGroup` modifica os parâmetros de um grupo de parâmetros de cluster de banco de dados.

Nível de acesso: `Write`.

Tipo de recurso: cluster-pg (obrigatório).

RDS: ClusterSnapshot atributo ModifyDB

`ModifyDBClusterSnapshotAttribute` adiciona um atributo e os valores ou remove um atributo e os valores de um snapshot do cluster de banco de dados manual.

Nível de acesso: `Write`.

Tipo de recurso: cluster-snapshot (obrigatório).

rds:ModifyDBInstance

`ModifyDBInstance` modifica as configurações de uma instância de banco de dados.

Nível de acesso: `Write`.

Ações dependentes: `iam:PassRole`.

Tipos de recurso:

- db (obrigatório).
- pg (obrigatório).

RDS: Modificar banco de dados ParameterGroup

`ModifyDBParameterGroup` modifica os parâmetros de um grupo de parâmetros de banco de dados.

Nível de acesso: `Write`.

Tipo de recurso: pg (obrigatório).

RDS: Modificar banco de dados SubnetGroup

`ModifyDBSubnetGroup` modifica um grupo de sub-redes de banco de dados existente.

Nível de acesso: `Write`.

Tipo de recurso: subgrp (obrigatório).

rds: Assinatura ModifyEvent

`ModifyEventSubscription` modifica uma assinatura de notificações de eventos existente do Neptune.

Nível de acesso: `Write`.

Tipo de recurso: es (obrigatório).

rds:RebootDBInstance

`RebootDBInstance` reinicia o serviço de mecanismo de banco de dados para a instância.

Nível de acesso: `Write`.

Tipo de recurso: db (obrigatório).

rds: RemoveRole FromDbCluster

`RemoveRoleFromDBCluster`dissocia uma função de AWS Identity and Access Management (IAM) de um cluster de banco de dados Amazon Neptune.

Nível de acesso: `Write`.

Ações dependentes: `iam:PassRole`.

Tipo de recurso: cluster (obrigatório).

rds: Assinatura RemoveSource IdentifierFrom

`RemoveSourceIdentifierFromSubscription` remove um identificador de origem de uma assinatura de notificações de eventos existente do Neptune.

Nível de acesso: `Write`.

Tipo de recurso: es (obrigatório).

vermelhos: RemoveTags FromResource

`RemoveTagsFromResource` remove as tags de metadados de um recurso do Neptune.

Nível de acesso: `Tagging`.

Tipos de recurso:

- cluster-snapshot
- db
- es
- pg
- subgrp

Chaves de condição:

- aws:RequestTag//tecla de *tag*
- leis: TagKeys

Grupo RDS: ResetDB ClusterParameter

`ResetDBClusterParameterGroup` modifica os parâmetros de um grupo de parâmetros de cluster de banco de dados para o valor padrão.

Nível de acesso: `Write`.

Tipo de recurso: cluster-pg (obrigatório).

RDS: redefinir banco de dados ParameterGroup

`ResetDBParameterGroup` modifica os parâmetros de um grupo de parâmetros de banco de dados para o valor padrão do mecanismo/sistema.

Nível de acesso: `Write`.

Tipo de recurso: pg (obrigatório).

RDS: instantâneo de banco de dados ClusterFrom restaurado

`RestoreDBClusterFromSnapshot` cria um cluster de banco de dados a partir de um snapshot de cluster de banco de dados.

Nível de acesso: `Write`.

Ações dependentes: `iam:PassRole`.

Tipos de recurso:

- cluster (obrigatório).
- cluster-snapshot (obrigatório).

Chaves de condição:

- aws:RequestTag//tecla de *tag*
- leis: TagKeys

RDS: tempo de banco de dados ClusterTo PointIn restaurado

`RestoreDBClusterToPointInTime` restaura um cluster de banco de dados para um momento arbitrário.

Nível de acesso: `Write`.

Ações dependentes: `iam:PassRole`.

Tipos de recurso:

- cluster (obrigatório).
- subgrp (obrigatório).

Chaves de condição:

- aws:RequestTag//tecla de *tag*
- leis: TagKeys

rds:StartDBCluster

`StartDBCluster` inicia o cluster de banco de dados especificado.

Nível de acesso: `Write`.

Tipo de recurso: cluster (obrigatório).

rds:StopDBCluster

`StopDBCluster` interrompe o cluster de banco de dados especificado.

Nível de acesso: `Write`.

Tipo de recurso: cluster (obrigatório).

Tipos de recurso disponíveis nas declarações de política administrativa do IAM do Neptune

O Neptune é compatível com os tipos de recurso na tabela a seguir para uso no elemento `Resource` das declarações de política de administração do IAM. Para obter mais informações sobre o elemento `Resource`, consulte Elementos de políticas JSON do IAM: recurso.

A list of Neptune administration actions identifica os tipos de recurso que podem ser especificados com cada ação. Um tipo de recurso também determina quais chaves de condição você pode incluir em uma política, como especificado na última coluna da tabela abaixo.

A coluna ARN na tabela abaixo especifica o nome do recurso da Amazon (ARN) que você deve usar para fazer referência a recursos desse tipo. As partes precedidas por um `$` devem ser substituídas pelos valores reais do cenário. Por exemplo, se você vir `$user-name` em um ARN, deverá substituir essa string pelo nome do usuário real do IAM ou por uma variável de política que contenha um nome do usuário do IAM. Para obter mais informações sobre ARNs, consulte ARNs do IAM e Trabalhar com ARNs administrativos no Amazon Neptune.

A coluna `Condition Keys` especifica chaves de contexto de condição que você pode incluir em uma declaração de política do IAM apenas quando esse recurso e uma ação de apoio compatível estão incluídos na declaração.

Tipos de recursos	ARN	Chaves de condição
`cluster` (um cluster de banco de dados)	`arn:`*`partition`*` :rds:`*`region`*`:`*`account-id`*` :cluster: `*`instance-name`*	aws:ResourceTag//tecla de *tag* rds:cluster-tag/*tag-key*
`cluster-pg` (um grupo de parâmetros do cluster de banco de dados)	`arn:`*`partition`*` :rds:`*`region`*`:`*`account-id`*` :cluster-pg: `*`neptune-DBClusterParameterGroupName`*	aws:ResourceTag//tecla de *tag*
`cluster-snapshot` (um snapshot de cluster de banco de dados)	`arn:`*`partition`*` :rds:`*`region`*`:`*`account-id`*` :cluster-snapshot: `*`neptune-DBClusterSnapshotName`*	aws:ResourceTag//tecla de *tag* rds:cluster-snapshot-tag/*tag-key*
db	`arn:`*`partition`*` :rds:`*`region`*`:`*`account-id`*` :db:`*`neptune-DbInstanceName`*	aws:ResourceTag//tecla de *tag*

Tipos de recursos	ARN	Chaves de condição
(uma instância de banco de dados)		vermelhos: DatabaseClass vermelhos: DatabaseEngine rds:db-tag/*tag-key*
es (uma assinatura de eventos)	arn:*partition* :rds:*region*:*account-id* :es:*neptune-CustSubscriptionId*	aws:ResourceTag//tecla de *tag* rds:es-tag/*tag-key*
pg (um grupo de parâmetros de banco de dados)	arn:*partition* :rds:*region*:*account-id* :pg:*neptune-ParameterGroupName*	aws:ResourceTag//tecla de *tag* rds:pg-tag/*tag-key*
subgrp (um grupo de sub-redes de banco de dados)	arn:*partition* :rds:*region*:*account-id* :subgrp:*neptune-DBSubnetGroupName* }	aws:ResourceTag//tecla de *tag* rds:subgrp-tag/*tag-key*

Chaves de condição disponíveis em declarações de política administrativa do IAM do Neptune

Usando chaves de condição, é possível especificar as condições em uma declaração de política do IAM para que a declaração só entre em vigor quando as condições forem verdadeiras. As chaves de condição que você pode usar nas declarações de política administrativa do Neptune se enquadram nas seguintes categorias:

- Chaves de condição globais — Elas são definidas por AWS para uso geral com AWS serviços. A maioria pode ser usada em declarações de política administrativa do Neptune.
- Chaves de condição de propriedades de recursos administrativos: essas chaves, listadas abaixo, são baseadas nas propriedades de recursos administrativos.
- Chaves de condição de acesso baseadas em tags: essas chaves, listadas abaixo, são baseadas em tags da AWS associadas aos recursos administrativos.

Chaves de condição de propriedades de recursos administrativos do Neptune

Chaves de condição	Descrição	Tipo
`rds:DatabaseClass`	Filtra o acesso pelo tipo de classe da instância de banco de dados.	String
`rds:DatabaseEngine`	Filtra o acesso pelo mecanismo do banco de dados Para valores possíveis, consulte o parâmetro do mecanismo na API CreateDBInstance	String
`rds:DatabaseName`	Filtra o acesso pelo nome definido pelo usuário do banco de dados na instância de banco de dados	String
`rds:EndpointType`	Filtra o acesso pelo tipo de endpoint Um dos seguintes: READER, WRITER, CUSTOM	String
`rds:Vpc`	Filtra o acesso pelo valor que especifica se a instância de banco de dados é executada em uma Amazon Virtual Private Cloud (Amazon VPC). Para indicar que a instância de banco de dados é executada em uma Amazon VPC, especifique `true`.	Booleano

Chaves de condição administrativas baseadas em tag

O Amazon Neptune é compatível com a especificação de condições em uma política do IAM usando tags personalizadas, para controlar o acesso ao Neptune por meio da Referência da API de gerenciamento.

Por exemplo, se você adicionar uma tag denominada `environment` às instâncias de banco de dados, com valores, como `beta`, `staging`, `production` e , depois, você poderá criar uma política que restrinja o acesso às instâncias com base no valor dessa tag.

> **Important**
>
> Se você gerenciar o acesso aos recursos do Neptune usando marcação, proteja o acesso às tags. Você pode restringir o acesso às tags criando políticas para as ações `AddTagsToResource` e `RemoveTagsFromResource`.
> Por exemplo, você pode usar a política a seguir para negar aos usuários a capacidade de adicionar ou remover tags para todos os recursos. Depois, você pode criar políticas para permitir que usuários específicos adicionem ou removam tags.
>
> ```
> { "Version": "2012-10-17",
>   "Statement":[
>     { "Sid": "DenyTagUpdates",
>       "Effect": "Deny",
>       "Action": [
>         "rds:AddTagsToResource",
>         "rds:RemoveTagsFromResource"
>       ],
>       "Resource":"*"
>     }
>   ]
> }
> ```

As chaves de condição baseadas em tags a seguir só funcionam com recursos administrativos em declarações de política administrativa.

Chaves de condição administrativas baseadas em tag

Chaves de condição	Descrição	Tipo
`aws:RequestTag/${TagKey}`	Filtra o acesso com base na presença de pares de chave-valor da tag na solicitação.	String

Chaves de condição	Descrição	Tipo
aws:ResourceTag/${TagKey}	Filtra o acesso com base nos pares de chave-valor da tag anexados ao recurso.	String
aws:TagKeys	Filtra o acesso com base na presença de chaves da tag na solicitação.	String
rds:cluster-pg-tag/${TagKey}	Filtra o acesso pela tag anexada a um grupo de parâmetros do cluster de banco de dados.	String
rds:cluster-snapshot-tag/${TagKey}	Filtra o acesso pela tag anexada a um snapshot do cluster de banco de dados.	String
rds:cluster-tag/${TagKey}	Filtra o acesso pela tag anexada a um cluster de banco de dados.	String
rds:db-tag/${TagKey}	Filtra o acesso pela tag anexada a uma instância de banco de dados.	String
rds:es-tag/${TagKey}	Filtra o acesso pela tag anexada a uma assinatura de evento.	String
rds:pg-tag/${TagKey}	Filtra o acesso pela tag anexada a um grupo de parâmetros de banco de dados.	String
rds:req-tag/${TagKey}	Filtra o acesso pelo conjunto de chaves da tag e valores que podem ser usados para marcar um recurso.	String
rds:secgrp-tag/${TagKey}	Filtra o acesso pela tag anexada a um grupo de segurança de banco de dados.	String

Chaves de condição	Descrição	Tipo
`rds:snapshot-tag/${TagKey}`	Filtra o acesso pela tag anexada a um snapshot de banco de dados.	String
`rds:subgrp-tag/${TagKey}`	Filtra o acesso pela etiqueta anexada a um grupo de sub-redes de banco de dados	String

Exemplos de declaração de política administrativa do IAM para o Neptune

Exemplos de política administrativa geral

Os exemplos a seguir mostram como criar políticas administrativas do Neptune que concedem permissões para realizar várias ações de gerenciamento em um cluster de banco de dados.

Política que impede que um usuário do IAM exclua uma instância de banco de dados especificada

Veja a seguir um exemplo de política que impede que um usuário do IAM exclua uma instância especificada do banco de dados Neptune:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyDeleteOneInstance",
      "Effect": "Deny",
      "Action": "rds:DeleteDBInstance",
      "Resource": "arn:aws:rds:us-west-2:123456789012:db:my-instance-name"
    }
  ]
}
```

Política que concede permissão para criar instâncias de banco de dados

Veja a seguir um exemplo de política que permite a um usuário do IAM criar instâncias de bancos de dados em um cluster de banco de dados especificado do Neptune:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCreateInstance",
      "Effect": "Allow",
      "Action": "rds:CreateDBInstance",
      "Resource": "arn:aws:rds:us-west-2:123456789012:cluster:my-cluster"
    }
  ]
}
```

Política que concede permissão para criar instâncias de banco de dados que usem um grupo de parâmetros de banco de dados específico

Veja a seguir um exemplo de política que permite a um usuário do IAM criar instâncias de banco de dados em um cluster de banco de dados especificado (aqui, `us-west-2`) em um cluster de banco de dados especificado do Neptune usando somente um grupo de parâmetros de banco de dados especificado.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCreateInstanceWithPG",
      "Effect": "Allow",
      "Action": "rds:CreateDBInstance",
      "Resource": [
        "arn:aws:rds:us-west-2:123456789012:cluster:my-cluster",
        "arn:aws:rds:us-west-2:123456789012:pg:my-instance-pg"
      ]
    }
  ]
}
```

Política que concede permissão para descrever qualquer recurso

Veja a seguir um exemplo de política que permite a um usuário do IAM descrever qualquer recurso do Neptune.

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowDescribe",
      "Effect": "Allow",
      "Action": "rds:Describe*",
      "Resource": *
    }
  ]
}
```

Exemplos de política administrativa baseada em tag

Os exemplos a seguir mostram como criar políticas administrativas do Neptune que concedam permissões para realizar várias ações de gerenciamento em um cluster de banco de dados.

Exemplo 1: conceder permissão para ações em um recurso usando uma tag personalizada que pode ter vários valores

A política abaixo permite o uso da API `ModifyDBInstance`, `CreateDBInstance` ou `DeleteDBInstance` em qualquer instância de banco de dados que tenha a tag `env` definida como `dev` ou `test`:

```
{ "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowDevTestAccess",
      "Effect": "Allow",
      "Action": [
        "rds:ModifyDBInstance",
        "rds:CreateDBInstance",
        "rds:DeleteDBInstance"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "rds:db-tag/env": [
              "dev",
              "test"
          ],
          "rds:DatabaseEngine": "neptune"
        }
      }
```

```
    }
  ]
}
```

Exemplo 2: Limitar o conjunto de chaves de tag e valores que podem ser usados para identificar um recurso

Esta política usa uma chave `Condition` para permitir que uma tag com a chave `env` e o valor `test`, `qa` ou `dev` seja adicionada a um recurso:

```
{ "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowTagAccessForDevResources",
      "Effect": "Allow",
      "Action": [
        "rds:AddTagsToResource",
        "rds:RemoveTagsFromResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "rds:req-tag/env": [
            "test",
            "qa",
            "dev"
          ],
          "rds:DatabaseEngine": "neptune"
        }
      }
    }
  ]
}
```

Exemplo 3: permitir acesso total aos recursos do Neptune com base em **`aws:ResourceTag`**

A seguinte política é semelhante ao primeiro exemplo acima, mas usa `aws:ResourceTag`:

```
{ "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowFullAccessToDev",
      "Effect": "Allow",
```

```
      "Action": [
        "rds:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/env": "dev",
          "rds:DatabaseEngine": "neptune"
        }
      }
    }
  ]
}
```

Declarações personalizadas de política de acesso aos dados do IAM para o Amazon Neptune

As declarações de política de acesso a dados do Neptune usam ações de acesso a dados, recursos e chaves de condição, todos precedidos por um prefixo `neptune-db:`.

Tópicos

- Usar ações de consulta nas declarações de política de acesso a dados do Neptune
- Usar ações em declarações de política de acesso a dados do IAM do Neptune
- Especificar recursos em declarações de política de acesso a dados do IAM no Neptune
- Chaves de condição disponíveis em declarações de política de acesso a dados do IAM do Neptune
- Exemplos de política de acesso a dados do IAM do Neptune

Usar ações de consulta nas declarações de política de acesso a dados do Neptune

Há três ações de consulta do Neptune que podem ser usadas em declarações de política de acesso a dados, a saber, `ReadDataViaQuery`, `WriteDataViaQuery` e `DeleteDataViaQuery`. Uma consulta específica pode precisar de permissões para realizar mais de uma dessas ações e nem sempre é óbvio qual combinação dessas ações deve ser permitida para executar uma consulta.

Antes de executar uma consulta, o Neptune determina as permissões necessárias para executar cada etapa da consulta e as combina no conjunto completo de permissões que a consulta exige. Observe que esse conjunto completo de permissões inclui todas as ações que a consulta pode

realizar, o que não é necessariamente o conjunto de ações que a consulta realmente executará quando for realizada em seus dados.

Isso significa que, para permitir que uma consulta específica seja executada, é necessário conceder permissões para cada ação que a consulta possa realizar, independentemente de ela realmente as executar ou não.

Veja alguns exemplos de consulta do Gremlin em que isso é explicado com mais detalhes:

- ```
  g.V().count()
  ```

  `g.V()` e `count()` exigem apenas acesso de leitura, portanto, a consulta como um todo exige apenas acesso `ReadDataViaQuery`.

- ```
  g.addV()
  ```

  `addV()` precisa conferir se um vértice com um ID específico existe ou não antes de inserir um novo. Isso significa que ele exige acesso `ReadDataViaQuery` e `WriteDataViaQuery`.

- ```
  g.V('1').as('a').out('created').addE('createdBy').to('a')
  ```

  `g.V('1').as('a')` e `out('created')` exigem apenas acesso de leitura, mas `addE().from('a')` requer acesso de leitura e gravação porque `addE()` precisa ler os vértices `from` e `to` e conferir se uma borda com o mesmo ID já existe antes de adicionar uma nova. A consulta como um todo, portanto, precisa de acesso `ReadDataViaQuery` e `WriteDataViaQuery`.

- ```
  g.V().drop()
  ```

  `g.V()` exige apenas acesso de leitura. `drop()` requer acesso de leitura e exclusão porque precisa ler um vértice ou uma borda antes de excluí-lo, portanto, a consulta como um todo exige acesso `ReadDataViaQuery` e `DeleteDataViaQuery`.

- ```
  g.V('1').property(single, 'key1', 'value1')
  ```

  `g.V('1')` requer apenas acesso de leitura, mas `property(single, 'key1', 'value1')` exige acesso de leitura, gravação e exclusão. Aqui, a etapa `property()` vai inserir a chave e o valor se eles ainda não existirem no vértice, mas se já existirem, ela vai excluir o valor da
  ```

propriedade existente e inserir um novo valor em seu lugar. Portanto, a consulta como um todo exige acesso `ReadDataViaQuery`, `WriteDataViaQuery` e `DeleteDataViaQuery`.

Qualquer consulta que contenha uma etapa `property()` precisará de permissões `ReadDataViaQuery`, `WriteDataViaQuery` e `DeleteDataViaQuery`.

Veja a seguir alguns exemplos do openCypher:

- ```
  MATCH (n)
  RETURN n
  ```

  Essa consulta lê todos os nós no banco de dados e os gera, o que exige apenas acesso `ReadDataViaQuery`.

- ```
  MATCH (n:Person)
  SET n.dept = 'AWS'
  ```

  Essa consulta exige acesso `ReadDataViaQuery`, `WriteDataViaQuery` e `DeleteDataViaQuery`. Ela lê todos os nós com o rótulo “Pessoa” e adiciona uma nova propriedade com a chave `dept` e o valor `AWS` a eles ou, se a propriedade `dept` já existir, vai excluir o valor antigo e inserir `AWS` em seu lugar. Além disso, se o valor a ser definido for `null`, `SET` excluirá completamente a propriedade.

  Como, em alguns casos, a cláusula `SET` talvez precise excluir um valor existente, ela sempre precisará de permissões `DeleteDataViaQuery`, além de permissões `ReadDataViaQuery` e `WriteDataViaQuery`.

- ```
  MATCH (n:Person)
  DETACH DELETE n
  ```

  Essa consulta precisa das permissões `ReadDataViaQuery` e `DeleteDataViaQuery`. Ela encontra todos os nós com o rótulo `Person` e os exclui junto com as bordas conectadas a esses nós e todas as propriedades e os rótulos associados.

- ```
  MERGE (n:Person {name: 'John'})-[:knows]->(:Person {name: 'Peter'})
  RETURN n
  ```

Essa consulta precisa das permissões `ReadDataViaQuery` e `WriteDataViaQuery`. A cláusula `MERGE` corresponde a um padrão especificado ou o cria. Como uma gravação poderá ocorrer se não houver correspondência ao padrão, são necessárias permissões de gravação e leitura.

Usar ações em declarações de política de acesso a dados do IAM do Neptune

Observe que as ações de acesso a dados do Neptune têm o prefixo `neptune-db:`, enquanto as ações administrativas no Neptune têm o prefixo `rds:`.

O nome do recurso da Amazon (ARN) de um recursos de dados no IAM não é o mesmo ARN atribuído ao cluster na criação. Você deve criar o ARN, conforme mostrado em Specifying data resources. Esses ARNs de recurso de dados podem usar curingas para incluir vários recursos.

As declarações de política de acesso a dados também podem incluir a chave de QueryLanguage condição neptune-db: para restringir o acesso por linguagem de consulta.

A partir da Versão: 1.2.0.0 (21/07/2022), o Neptune é compatível com a restrição de permissões a uma ou mais ações específicas do Neptune. Isso oferece um controle de acesso mais detalhado do que era possível anteriormente.

Important

- As alterações em uma política do IAM demoram até dez minutos para ser aplicadas aos recursos do Neptune especificados.
- As políticas do IAM aplicadas a um cluster de banco de dados do Neptune são aplicadas a todas as instâncias desse cluster.

Ações de acesso a dados baseadas em consulta

Note

Nem sempre é óbvio quais permissões são necessárias para executar uma consulta específica, porque as consultas podem realizar mais de uma ação, dependendo dos dados processados. Consulte Usar ações de consulta Para mais informações.

neptune-db:ReadDataViaQuery

`ReadDataViaQuery` possibilita ao usuário ler dados do banco de dados Neptune enviando consultas.

Grupos de ações: somente leitura, leitura e gravação.

Chaves de contexto de ação: `neptune-db:QueryLanguage`.

Recursos necessários: banco de dados.

neptune-db:WriteDataViaQuery

`WriteDataViaQuery` possibilita ao usuário gravar dados no banco de dados Neptune enviando consultas.

Grupos de ações: leitura e gravação.

Chaves de contexto de ação: `neptune-db:QueryLanguage`.

Recursos necessários: banco de dados.

neptune-db:DeleteDataViaQuery

`DeleteDataViaQuery` possibilita ao usuário excluir dados do banco de dados Neptune enviando consultas.

Grupos de ações: leitura e gravação.

Chaves de contexto de ação: `neptune-db:QueryLanguage`.

Recursos necessários: banco de dados.

neptune-db:GetQueryStatus

`GetQueryStatus` possibilita ao usuário conferir o status de todas as consultas ativas.

Grupos de ações: somente leitura, leitura e gravação.

Chaves de contexto de ação: `neptune-db:QueryLanguage`.

Recursos necessários: banco de dados.

neptune-db:GetStreamRecords

`GetStreamRecords` possibilita ao usuário buscar registros de fluxo do Neptune.

Grupos de ações: leitura e gravação.

Chaves de contexto de ação: `neptune-db:QueryLanguage`.

Recursos necessários: banco de dados.

neptune-db:CancelQuery

`CancelQuery` possibilita ao usuário cancelar uma consulta.

Grupos de ações: leitura e gravação.

Recursos necessários: banco de dados.

Ações gerais de acesso a dados

neptune-db:GetEngineStatus

`GetEngineStatus` possibilita ao usuário conferir o status do mecanismo do Neptune.

Grupos de ações: somente leitura, leitura e gravação.

Recursos necessários: banco de dados.

neptune-db:GetStatisticsStatus

`GetStatisticsStatus` possibilita ao usuário conferir o status das estatísticas que estão sendo coletadas para o banco de dados.

Grupos de ações: somente leitura, leitura e gravação.

Recursos necessários: banco de dados.

neptune-db:GetGraphSummary

`GetGraphSummary` A API de resumo do grafo possibilita a você recuperar um resumo somente leitura do grafo.

Grupos de ações: somente leitura, leitura e gravação.

Recursos necessários: banco de dados.

neptune-db:ManageStatistics

`ManageStatistics` possibilita ao usuário gerenciar a coleta de estatísticas para o banco de dados.

Grupos de ações: leitura e gravação.

Recursos necessários: banco de dados.

neptune-db:DeleteStatistics

`DeleteStatistics` permite ao usuário excluir todas as estatísticas no banco de dados.

Grupos de ações: leitura e gravação.

Recursos necessários: banco de dados.

neptune-db:ResetDatabase

`ResetDatabase` possibilita ao usuário obter o token necessário para uma redefinição e redefinir o banco de dados Neptune.

Grupos de ações: leitura e gravação.

Recursos necessários: banco de dados.

Ações de acesso a dados do carregador em massa

neptune-db:StartLoaderJob

`StartLoaderJob` possibilita ao usuário iniciar um trabalho de carregador em massa.

Grupos de ações: leitura e gravação.

Recursos necessários: banco de dados.

neptune-db:GetLoaderJobStatus

`GetLoaderJobStatus` possibilita ao usuário conferir o status de um trabalho de carregador em massa.

Grupos de ações: somente leitura, leitura e gravação.

Recursos necessários: banco de dados.

neptune-db:ListLoaderJobs

`ListLoaderJobs` possibilita ao usuário listar todos os trabalhos de carregador em massa.

Grupos de ações: somente lista, somente leitura e gravação e leitura.

Recursos necessários: banco de dados.

neptune-db:CancelLoaderJob

`CancelLoaderJob` possibilita ao usuário cancelar um trabalho de carregador.

Grupos de ações: leitura e gravação.

Recursos necessários: banco de dados.

Ações de acesso a dados de machine learning

neptune-db:StartMLDataProcessingJob

`StartMLDataProcessingJob` possibilita a um usuário iniciar um trabalho de processamento de dados do Neptune ML.

Grupos de ações: leitura e gravação.

Recursos necessários: banco de dados.

neptune-db:StartMLModelTrainingJob

`StartMLModelTrainingJob` possibilita a um usuário iniciar um trabalho de treinamento de modelo de ML.

Grupos de ações: leitura e gravação.

Recursos necessários: banco de dados.

neptune-db:StartMLModelTransformJob

`StartMLModelTransformJob` possibilita a um usuário iniciar um trabalho de transformação de modelo de ML.

Grupos de ações: leitura e gravação.

Recursos necessários: banco de dados.

neptune-db:CreateMLEndpoint

`CreateMLEndpoint` possibilita a um usuário criar um endpoint do Neptune ML.

Grupos de ações: leitura e gravação.

Recursos necessários: banco de dados.

neptune-db:GetMLDataProcessingJobStatus

`GetMLDataProcessingJobStatus` possibilita a um usuário conferir o status de um trabalho de processamento de dados do Neptune ML.

Grupos de ações: somente leitura, leitura e gravação.

Recursos necessários: banco de dados.

neptune-db:GetMLModelTrainingJobStatus

`GetMLModelTrainingJobStatus` possibilita a um usuário conferir o status de um trabalho de treinamento de modelos do Neptune ML.

Grupos de ações: somente leitura, leitura e gravação.

Recursos necessários: banco de dados.

neptune-db:GetMLModelTransformJobStatus

`GetMLModelTransformJobStatus` possibilita a um usuário conferir o status de um trabalho de transformação de modelos do Neptune ML.

Grupos de ações: somente leitura, leitura e gravação.

Recursos necessários: banco de dados.

neptune-db:GetMLEndpointStatus

`GetMLEndpointStatus` possibilita a um usuário conferir o status de um endpoint do Neptune ML.

Grupos de ações: somente leitura, leitura e gravação.

Recursos necessários: banco de dados.

neptune-db:ListMLDataProcessingJobs

`ListMLDataProcessingJobs` possibilita a um usuário listar trabalhos de processamento de dados do Neptune ML.

Grupos de ações: somente lista, somente leitura e gravação e leitura.

Recursos necessários: banco de dados.

neptune-db:ListMLModelTrainingJobs

`ListMLModelTrainingJobs` possibilita a um usuário listar todos os trabalhos de treinamento de modelos do Neptune ML.

Grupos de ações: somente lista, somente leitura e gravação e leitura.

Recursos necessários: banco de dados.

neptune-db:ListMLModelTransformJobs

`ListMLModelTransformJobs` possibilita a um usuário listar todos os trabalhos de transformação de modelos de ML.

Grupos de ações: somente lista, somente leitura e gravação e leitura.

Recursos necessários: banco de dados.

neptune-db:ListMLEndpoints

`ListMLEndpoints` possibilita a um usuário listar todos os endpoints do Neptune ML.

Grupos de ações: somente lista, somente leitura e gravação e leitura.

Recursos necessários: banco de dados.

neptune-db:CancelMLDataProcessingJob

`CancelMLDataProcessingJob` possibilita a um usuário cancelar um trabalho de processamento de dados do Neptune ML.

Grupos de ações: leitura e gravação.

Recursos necessários: banco de dados.

neptune-db:CancelMLModelTrainingJob

`CancelMLModelTrainingJob` possibilita a um usuário cancelar um trabalho de treinamento de modelos do Neptune ML.

Grupos de ações: leitura e gravação.

Recursos necessários: banco de dados.

`neptune-db:CancelMLModelTransformJob`

`CancelMLModelTransformJob` possibilita a um usuário cancelar um trabalho de transformação de modelos do Neptune ML.

Grupos de ações: leitura e gravação.

Recursos necessários: banco de dados.

`neptune-db:DeleteMLEndpoint`

`DeleteMLEndpoint` possibilita a um usuário excluir um endpoint do Neptune ML.

Grupos de ações: leitura e gravação.

Recursos necessários: banco de dados.

Especificar recursos em declarações de política de acesso a dados do IAM no Neptune

Recursos de dados, como ações de dados, têm um prefixo `neptune-db:`.

Em uma política de acesso a dados do Neptune, você especifica o cluster de banco de dados ao qual está concedendo acesso a um ARN com o seguinte formato:

```
arn:aws:neptune-db:region:account-id:cluster-resource-id/*
```

Esse ARN de recurso contém as seguintes partes:

- *`region`*é a AWS região do cluster de banco de dados Amazon Neptune.
- *`account-id`* é o número da conta da AWS do cluster de banco de dados.
- *`cluster-resource-id`* é um ID de recurso do cluster de banco de dados.

> **Important**
>
> O `cluster-resource-id` é diferente do identificador do cluster. Para encontrar um ID de recurso de cluster no AWS Management Console Neptune, procure na seção Configuração o cluster de banco de dados em questão.

Chaves de condição disponíveis em declarações de política de acesso a dados do IAM do Neptune

Usando chaves de condição, é possível especificar as condições em uma declaração de política do IAM para que a declaração só entre em vigor quando as condições forem verdadeiras.

As chaves de condição que você pode usar em declarações de política de acesso a dados do Neptune se enquadram nas seguintes categorias:

- Chaves de condição globais — O subconjunto de chaves de condição AWS globais que o Neptune suporta nas declarações de política de acesso a dados está listado abaixo.
- Chaves de condição específicas do serviço: são chaves definidas pelo Neptune especificamente para uso em declarações de política de acesso a dados. No momento, há apenas um, neptune-db:QueryLanguage, que concede acesso somente se uma linguagem de consulta específica estiver sendo usada.

AWS chaves de contexto de condição global suportadas pelo Neptune em declarações de política de acesso a dados

A seguinte tabela lista o subconjunto de chaves de contexto de condição globais da AWS aceitas pelo Amazon Neptune para uso em declarações de política de acesso a dados:

Chaves de condição globais que você pode usar em declarações de política de acesso a dados

Chaves de condição	Descrição	Tipo
`aws:CurrentTime`	Filtra o acesso pela data e hora atuais da solicitação.	`String`
`aws:EpochTime`	Filtra o acesso pela data e hora da solicitação expressa como um valor de época do UNIX.	`Numeric`
`aws:PrincipalAccount`	Filtra o acesso pela conta à qual a entidade principal solicitante pertence.	`String`
`aws:PrincipalArn`	Filtra o acesso pelo ARN da entidade principal que faz a solicitação.	`String`

Chaves de condição	Descrição	Tipo
aws:PrincipalIsAWSService	Permite acesso somente se a chamada estiver sendo feita diretamente por um responsável pelo AWS serviço.	Boolean
aws:PrincipalOrgID	Filtra o acesso pelo identificador da organização em AWS Organizations à qual o principal solicitante pertence.	String
aws:PrincipalOrgPaths	Filtra o acesso pelo caminho AWS Organizations para o diretor que está fazendo a solicitação.	String
aws:PrincipalTag	Filtra o acesso por uma tag associada à entidade principal que faz a solicitação.	String
aws:PrincipalType	Filtra o acesso pelo tipo de entidade principal que faz a solicitação.	String
aws:RequestedRegion	Filtra o acesso pela AWS região que foi chamada na solicitação.	String
aws:SecureTransport	Viabilizará o acesso somente se a solicitação tiver sido enviada usando SSL.	Boolean
aws:SourceIp	Filtra o acesso pelo endereço IP do solicitante.	String
aws:TokenIssueTime	Filtra o acesso pela data e a hora em que as credenciais de segurança temporárias foram emitidas.	String
aws:UserAgent	Filtra o acesso pelo aplicativo cliente do solicitante.	String
aws:userid	Filtra o acesso pelo identificador da entidade principal do solicitante.	String

Chaves de condição	Descrição	Tipo
aws:ViaAWSService	Permite acesso somente se um AWS serviço tiver feito a solicitação em seu nome.	Boolean

Chaves de condição específicas do serviço do Neptune

O Neptune é compatível com a seguinte chave de condição específica do serviço para políticas do IAM:

Chaves de condição específicas do serviço do Neptune

Chaves de condição	Descrição	Tipo
neptune-db:QueryLanguage	Filtra o acesso aos dados pela linguagem de consulta que está sendo usada. Os valores válidos são: Gremlin, OpenCypher e Sparql. As ações compatíveis são ReadDataViaQuery , WriteDataViaQuery , DeleteDataViaQuery , GetQueryStatus e CancelQuery .	String

Exemplos de política de acesso a dados do IAM do Neptune

Os seguintes exemplos mostram como criar políticas personalizadas do IAM que usem controle de acesso refinado de APIs de planos de dados e ações, introduzidas na versão 1.2.0.0 do mecanismo do Neptune.

Exemplo de política que viabiliza acesso irrestrito aos dados em um cluster de banco de dados do Neptune

O exemplo de política a seguir permite que um usuário do IAM se conecte a um cluster de banco de dados do Neptune usando a autenticação de banco de dados do IAM e usa o caractere “*“ para corresponder a todas as ações disponíveis.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "neptune-db:*",
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

O exemplo anterior inclui um ARN de recurso em um formato específico da autenticação do IAM no Neptune. Para criar o ARN, consulte Specifying data resources. Observe que o ARN usando para um `Resource` de autorização do IAM não é o mesmo ARN atribuído ao cluster na criação.

Exemplo de política que viabiliza acesso somente leitura a um cluster de banco de dados do Neptune

A política a seguir concede permissão para acesso total somente leitura aos dados em um cluster de banco de dados do Neptune:

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect": "Allow",
      "Action": [
        "neptune-db:Read*",
        "neptune-db:Get*",
        "neptune-db:List*"
      ],
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Exemplo de política que nega acesso todo o acesso a um cluster de banco de dados do Neptune

A ação padrão do IAM é negar o acesso a um cluster de banco de dados a menos que um Effect `Allow` seja concedido. No entanto, a política a seguir nega todo o acesso a um cluster de banco

de dados para uma determinada AWS conta e região, que então tem precedência sobre qualquer `Allow` efeito.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "neptune-db:*",
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Exemplo de política que concede acesso de leitura por meio de consultas

A seguinte política apenas concede permissão para leitura de um cluster de banco de dados do Neptune usando uma consulta:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "neptune-db:ReadDataViaQuery",
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Exemplo de política que só permite consultas do Gremlin

A seguinte política usa a chave de condição `neptune-db:QueryLanguage` para conceder permissão para consultar o Neptune somente usando a linguagem de consulta Gremlin:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Action": [
        "neptune-db:ReadDataViaQuery",
        "neptune-db:WriteDataViaQuery",
        "neptune-db:DeleteDataViaQuery"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "neptune-db:QueryLanguage": "Gremlin"
        }
      }
    }
  ]
}
```

Exemplo de política que permite todo o acesso, exceto ao gerenciamento de modelos do Neptune ML

A seguinte política concede acesso total às operações de grafo do Neptune, exceto os atributos de gerenciamento de modelos do Neptune ML:

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect": "Allow",
      "Action": [
        "neptune-db:CancelLoaderJob",
        "neptune-db:CancelQuery",
        "neptune-db:DeleteDataViaQuery",
        "neptune-db:DeleteStatistics",
        "neptune-db:GetEngineStatus",
        "neptune-db:GetLoaderJobStatus",
        "neptune-db:GetQueryStatus",
        "neptune-db:GetStatisticsStatus",
        "neptune-db:GetStreamRecords",
        "neptune-db:ListLoaderJobs",
        "neptune-db:ManageStatistics",
        "neptune-db:ReadDataViaQuery",
        "neptune-db:ResetDatabase",
        "neptune-db:StartLoaderJob",
        "neptune-db:WriteDataViaQuery"
      ],
```

```
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Exemplo de política que permite acesso ao gerenciamento de modelos do Neptune ML

Essa política concede acesso aos atributos de gerenciamento de modelos do Neptune ML:

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect": "Allow",
      "Action": [
        "neptune-db:CancelMLDataProcessingJob",
        "neptune-db:CancelMLModelTrainingJob",
        "neptune-db:CancelMLModelTransformJob",
        "neptune-db:CreateMLEndpoint",
        "neptune-db:DeleteMLEndpoint",
        "neptune-db:GetMLDataProcessingJobStatus",
        "neptune-db:GetMLEndpointStatus",
        "neptune-db:GetMLModelTrainingJobStatus",
        "neptune-db:GetMLModelTransformJobStatus",
        "neptune-db:ListMLDataProcessingJobs",
        "neptune-db:ListMLEndpoints",
        "neptune-db:ListMLModelTrainingJobs",
        "neptune-db:ListMLModelTransformJobs",
        "neptune-db:StartMLDataProcessingJob",
        "neptune-db:StartMLModelTrainingJob",
        "neptune-db:StartMLModelTransformJob"
      ],
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Exemplo de política que concede acesso total por consulta

A seguinte política concede acesso total às operações de consulta de grafo do Neptune, mas não a atributos, como redefinição rápida, fluxos, carregador em massa, gerenciamento de modelos do Neptune ML, etc.:

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect": "Allow",
      "Action": [
        "neptune-db:ReadDataViaQuery",
        "neptune-db:WriteDataViaQuery",
        "neptune-db:DeleteDataViaQuery",
        "neptune-db:GetEngineStatus",
        "neptune-db:GetQueryStatus",
        "neptune-db:CancelQuery"
      ],
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Exemplo de política que concede acesso total somente para consultas do Gremlin

A seguinte política concede acesso total às operações de consulta de grafo do Neptune usando a linguagem de consulta Gremlin, mas não a consultas em outras linguagens e não a atributos, como redefinição rápida, fluxos, carregador em massa, gerenciamento de modelos do Neptune ML, etc.:

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect": "Allow",
      "Action": [
        "neptune-db:ReadDataViaQuery",
        "neptune-db:WriteDataViaQuery",
        "neptune-db:DeleteDataViaQuery",
        "neptune-db:GetEngineStatus",
        "neptune-db:GetQueryStatus",
```

```
        "neptune-db:CancelQuery"
      ],
      "Resource": [
        "arn:aws:neptune-db:us-east-1:123456789012:cluster-ABCD1234EFGH5678IJKL90MNOP/
*"
      ],
      "Condition": {
        "StringEquals": {
           "neptune-db:QueryLanguage":"Gremlin"
        }
      }
    }
  ]
}
```

Exemplo de política que concede acesso total, exceto para redefinição rápida

A seguinte política concede acesso total a um cluster de banco de dados do Neptune, exceto para uso de redefinição rápida:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "neptune-db:*",
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    },
    {
      "Effect": "Deny",
      "Action": "neptune-db:ResetDatabase",
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Usar perfis vinculados a serviços para Neptune

O Amazon Neptune AWS Identity and Access Management usa funções vinculadas a serviços (IAM). A função vinculada ao serviço é um tipo exclusivo de perfil do IAM vinculado diretamente

ao Neptune. As funções vinculadas ao serviço são predefinidas pelo Neptune e incluem todas as permissões que o serviço exige para chamar outros serviços em seu nome. AWS

> **Important**
>
> Para determinados atributos de gerenciamento, o Amazon Neptune usa tecnologia operacional compartilhada com o Amazon RDS. Isso inclui a Função vinculada ao serviço e permissões da API de gerenciamento.

Um perfil vinculado ao serviço facilita a configuração do Neptune porque você não precisa adicionar as permissões necessárias manualmente. O Neptune define as permissões dos perfis vinculados a serviços e, exceto se definido de outra forma, somente o Neptune pode assumir os perfis. As permissões definidas incluem a política de confiança e a política de permissões, e essa política não pode ser anexada a nenhuma outra entidade do IAM.

É possível excluir as funções somente depois de primeiro excluir seus recursos relacionados. Isso protege os recursos do Neptune, pois você não pode remover por engano as permissões para acessar os recursos.

Para obter informações sobre outros serviços compatíveis com funções vinculadas a serviços, consulte AWS Services That Work with IAM e procure os serviços que contêm Yes na coluna Service-Linked Role. Escolha um Sim com um link para visualizar a documentação da função vinculada a esse serviço.

Permissões de perfil vinculado ao serviço para Neptune

O Neptune usa `AWSServiceRoleForRDS` a função vinculada ao serviço para permitir que o Neptune e o Amazon AWS RDS chamem serviços em nome de suas instâncias de banco de dados. A função vinculada ao serviço `AWSServiceRoleForRDS` confia no serviço `rds.amazonaws.com` para presumir a função.

A política de permissões do perfil permite que o Neptune conclua as seguintes ações nos recursos especificados:

- Ações em `ec2`:
  - `AssignPrivateIpAddresses`
  - `AuthorizeSecurityGroupIngress`

  - `CreateNetworkInterface`
  - `CreateSecurityGroup`
  - `DeleteNetworkInterface`
  - `DeleteSecurityGroup`
  - `DescribeAvailabilityZones`
  - `DescribeInternetGateways`
  - `DescribeSecurityGroups`
  - `DescribeSubnets`
  - `DescribeVpcAttribute`
  - `DescribeVpcs`
  - `ModifyNetworkInterfaceAttribute`
  - `RevokeSecurityGroupIngress`
  - `UnassignPrivateIpAddresses`
- Ações em `sns`:
  - `ListTopic`
  - `Publish`
- Ações em `cloudwatch`:
  - `PutMetricData`
  - `GetMetricData`
  - `CreateLogStream`
  - `PullLogEvents`
  - `DescribeLogStreams`
  - `CreateLogGroup`

> **Note**
>
> É necessário configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou função) crie, edite ou exclua um perfil vinculado ao serviço. Você pode encontrar a seguinte mensagem de erro:
> Impossível criar o recurso. Você se você tem permissão para criar o perfil vinculado ao serviço. Caso contrário, aguarde e tente novamente mais tarde.

Se essa mensagem for exibida, verifique se você tem as seguintes permissões habilitadas:

```
{
    "Action": "iam:CreateServiceLinkedRole",
    "Effect": "Allow",
    "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
    "Condition": {
        "StringLike": {
            "iam:AWSServiceName":"rds.amazonaws.com"
        }
    }
}
```

Para mais informações, consulte Permissões de perfil vinculado ao serviço no Guia do usuário do IAM.

Criar um perfil vinculado ao serviço para Neptune

Não é necessário criar manualmente uma função vinculada ao serviço. Quando você cria uma instância ou um cluster, o Neptune cria um perfil vinculado a serviço para você.

Important

Para saber mais, consulte A New Role Appeared in My IAM Account no Guia do usuário do IAM.

Se você excluir essa função vinculada ao serviço e precisar criá-la novamente, poderá usar esse mesmo processo para recriar a função em sua conta. Quando você cria uma instância ou um cluster, o Neptune cria o perfil vinculado ao serviço novamente.

Editar um perfil vinculado ao serviço para Neptune

O Neptune não permite editar o perfil vinculada ao serviço `AWSServiceRoleForRDS`. Depois que criar um perfil vinculado ao serviço, você não poderá alterar o nome do perfil, pois várias entidades podem fazer referência a ele. No entanto, será possível editar a descrição do perfil usando o IAM. Para obter mais informações, consulte Editar uma função vinculada a serviço no Guia do usuário do IAM.

Excluir um perfil vinculado ao serviço para Neptune

Se você não precisar mais usar um atributo ou serviço que requer uma função vinculada a serviço, é recomendável excluí-la. Dessa forma, você não tem uma entidade não utilizada que não seja monitorada ativamente ou mantida. Contudo, você deve excluir todas as suas instâncias e clusters para poder excluir a função vinculada a serviço associada.

Limpeza de uma função vinculada a serviço antes da exclusão

Antes de você poder usar o IAM para excluir uma função vinculada ao serviço, você deve primeiro confirmar que a função não tem sessões ativas e remover quaisquer recursos usados pela função.

Para verificar se a função vinculada ao serviço tem uma sessão ativa no console do IAM

1. Faça login AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação do console do IAM, escolha Funções. A seguir, selecione o nome (não a caixa de seleção) da função `AWSServiceRoleForRDS`.
3. Na página Resumo para a função selecionada, escolha a guia Consultor de Acesso.
4. Na guia Consultor de Acesso, revise a atividade recente para a função vinculada ao serviço.

> **Note**
>
> Se não tiver certeza se o Neptune está usando o perfil `AWSServiceRoleForRDS`, você poderá tentar excluir o perfil. Se o serviço estiver usando a função, a exclusão falhará e você poderá visualizar as regiões da em que a função está sendo usada. Se a função está sendo usada, você deve aguardar a sessão final antes de excluir a função. Não é possível revogar a sessão de uma função vinculada a um serviço.

Se deseja remover a função `AWSServiceRoleForRDS`, você deve primeiro excluir todas as suas instâncias e clusters.

Exclusão de todas as instâncias

Use um destes procedimentos para excluir cada uma de suas instâncias.

Para excluir uma instância (console)

1. Abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No painel de navegação, escolha Instâncias.
3. Na lista Instances, escolha a instância que você deseja excluir.
4. Escolha Instance actions e, em seguida, escolha Delete.
5. Se for exibido Create final Snapshot? (Criar snapshot final?), escolha Yes (Sim) ou No (Não).
6. Se você escolher Yes (Sim) na etapa anterior, em Final snapshot name (Nome do snapshot final), digite o nome do snapshot final.
7. Escolha Excluir.

Como excluir uma instância (AWS CLI)

Consulte `delete-db-instance` no AWS CLI Command Reference.

Para excluir uma instância (API)

Consulte `DeleteDBInstance`.

Exclusão de todos os clusters

Siga um destes procedimentos para excluir um único cluster e, depois, repita o procedimento para cada um dos seus clusters.

Para excluir um cluster (console)

1. Faça login no AWS Management Console e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. Na lista Clusters, escolha o cluster que você deseja excluir.
3. Escolha Ações de cluster e Excluir.
4. Escolha Excluir.

Para excluir um cluster (CLI)

Consulte `delete-db-cluster` no AWS CLI Command Reference.

Para excluir um cluster (API)

Consulte `DeleteDBCluster`

É possível usar o console, a CLI ou a API do IAM para excluir o perfil vinculado ao serviço `AWSServiceRoleForRDS`. Para mais informações, consulte Excluir um perfil vinculado ao serviço no Guia do usuário do IAM.

Autenticação do IAM usando credenciais temporárias

O Amazon Neptune é compatível com a autenticação do IAM usando credenciais temporárias.

Você pode usar uma função assumida para autenticar ao usar uma política de autenticação do IAM como uma das políticas de exemplo nas seções anteriores.

Se estiver usando credenciais temporárias, você deve especificar `AWS_SESSION_TOKEN` bem como `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, e `SERVICE_REGION`.

> **Note**
>
> As credenciais temporárias expiram após um intervalo especificado, incluindo o token de sessão.
> Você deve atualizar o seu token de sessão ao solicitar novas credenciais. Para obter mais informações, consulte Usando credenciais de segurança temporárias para solicitar acesso aos AWS recursos.

As seções a seguir descrevem como permitir o acesso e recuperar credenciais temporárias.

Para autenticar usando credenciais temporárias

1. Criar um perfil do IAM com permissão para acessar um cluster do Neptune. Para obter informações sobre como criar essa função, consulte the section called “Tipos de política do IAM”.
2. Adicionar um relacionamento de confiança à função que permita acesso as credenciais.

   Recupere as credenciais temporárias, incluindo `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, e `AWS_SESSION_TOKEN`.
3. Conecte-se ao cluster do Neptune e assine as solicitações usando as credenciais temporárias. Para obter mais informações sobre como conectar e assinar solicitações, consulte the section called “Conexão e assinatura”.

Há vários métodos para recuperar credenciais temporárias dependendo do ambiente.

Tópicos

- Como obter credenciais temporárias com a AWS CLI
- Configurando o AWS Lambda para a autenticação do Neptune IAM
- Configurar o Amazon EC2 para autenticação do IAM do Neptune

Como obter credenciais temporárias com a AWS CLI

Para obter credenciais usando o AWS Command Line Interface (AWS CLI), primeiro você precisa adicionar uma relação de confiança que conceda permissão para assumir a função ao AWS usuário que executará o AWS CLI comando.

Adicione o seguinte relacionamento de confiança ao perfil de autenticação do IAM do Neptune. Se você não tiver um perfil de autenticação do IAM do Neptune, consulte the section called “Tipos de política do IAM”.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/test"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Para obter informações sobre como ampliar a relação de confiança à função, consulte Editar a relação de confiança de uma função existente no Guia de administração do AWS Directory Service .

Se a política do Neptune ainda não estiver associada a um perfil, crie um perfil. Associe a política de autenticação do IAM do Neptune e, depois, adicione a política de confiança. Para obter informações sobre como criar uma nova função, consulte Como criar uma função.

> **Note**
>
> As seções a seguir pressupõem que você tenha o AWS CLI instalado.

Para executar o AWS CLI manualmente

1. Digite o comando a seguir para solicitar as credenciais usando a AWS CLI. Substitua o ARN da função, nome de sessão e perfil com seus próprios valores.

```
aws sts assume-role  --role-arn  arn:aws:iam::123456789012:role/NeptuneIAMAuthRole
 --role-session-name test --profile testprofile
```

2. A seguir está um exemplo de saída do comando. A seção `Credentials` contém os valores de que você precisa.

> **Note**
>
> Anote o valor `Expiration`, pois ele será necessário para obter novas credenciais após esse período.

```
{
    "AssumedRoleUser": {
        "AssumedRoleId": "AROA3XFRBF535PLBIFPI4:s3-access-example",
        "Arn": "arn:aws:sts::123456789012:assumed-role/xaccounts3access/s3-access-
example"
    },
    "Credentials": {
        "SecretAccessKey": "9drTJvcXLB89EXAMPLELB8923FB892xMFI",
        "SessionToken": "AQoXdzELDDY//////////
wEaoAK1wvxJY12r2IrDFT2IvAzTCn3zHoZ7YNtpiQLF0MqZye/qwjzP2iEXAMPLEbw/
m3hsj8VBTkPORGvr9jM5sgP+w9IZWZnU+LWhmg
+a5fDi2oTGUYcdg9uexQ4mtCHIHfi4citgqZTgco40Yqr4lIlo4V2b2Dyauk0eYFNebHtYlFVgAUj
+7Indz3LU0aTWk1WKIjHmmMCIoTkyYp/k7kUG7moeEYKSitwQIi6Gjn+nyzM
+PtoA3685ixzv0R7i5rjQi0YE0lf1oeie3bDiNHncmzosRM6SFiPzSvp6h/32xQuZsjcypmwsPSDtTPYcs0+YN/8BRi
IcrxSpnWEXAMPLEXSDFTAQAM6Dl9zR0tXoybnlrZIwMLlMi1Kcgo5OytwU=",
        "Expiration": "2016-03-15T00:05:07Z",
        "AccessKeyId": "ASIAJEXAMPLEXEG2JICEA"
    }
```

```
}
```

3. Defina as variáveis do ambiente usando as credenciais retornadas.

```
export AWS_ACCESS_KEY_ID=ASIAJEXAMPLEXEG2JICEA
export AWS_SECRET_ACCESS_KEY=9drTJvcXLB89EXAMPLELB8923FB892xMFI
export AWS_SESSION_TOKEN=AQoXdzELDDY//////////
wEaoAK1wvxJY12r2IrDFT2IvAzTCn3zHoZ7YNtpiQLF0MqZye/qwjzP2iEXAMPLEbw/
m3hsj8VBTkPORGvr9jM5sgP+w9IZWZnU+LWhmg
+a5fDi2oTGUYcdg9uexQ4mtCHIHfi4citgqZTgco40Yqr4lIlo4V2b2Dyauk0eYFNebHtYlFVgAUj
+7Indz3LU0aTWk1WKIjHmmMCIoTkyYp/k7kUG7moeEYKSitwQIi6Gjn+nyzM
+PtoA3685ixzv0R7i5rjQi0YE0lf1oeie3bDiNHncmzosRM6SFiPzSvp6h/32xQuZsjcypmwsPSDtTPYcs0+YN/8BRi
IcrxSpnWEXAMPLEXSDFTAQAM6Dl9zR0tXoybnlrZIwMLlMi1Kcgo5OytwU=

export SERVICE_REGION=us-east-1 or us-east-2 or us-west-1 or us-west-2 or ca-
central-1 or
                      sa-east-1 or eu-north-1 or eu-west-1 or eu-west-2 or eu-
west-3 or eu-central-1 or me-south-1 or
                      me-central-1 or il-central-1 or af-south-1 or ap-east-1 or
 ap-northeast-1 or ap-northeast-2 or ap-southeast-1 or ap-southeast-2 or ap-south-1
 or
                      cn-north-1 or cn-northwest-1 or
                      us-gov-east-1 or us-gov-west-1
```

4. Conectar-se usando um dos seguintes métodos.

   - the section called “Console do Gremlin”
   - the section called “Gremlin Java”
   - the section called “SPARQL Java (RDF4J e Jena)”
   - the section called “Exemplo Python”

Para usar um script para obter as credenciais

1. Execute o seguinte comando para instalar comando jq. O script usa esse comando para analisar a saída do AWS CLI comando.

```
sudo yum -y install jq
```

2. Crie um arquivo chamado `credentials.sh` em um editor de texto e adicione o texto a seguir. Substitua a região de serviço, ARN da função, nome de sessão e perfil com seus próprios valores.

```
#!/bin/bash

creds_json=$(aws sts assume-role  --role-arn  arn:aws:iam::123456789012:role/
NeptuneIAMAuthRole --role-session-name test --profile testprofile)

export AWS_ACCESS_KEY_ID=$(echo "$creds_json" | jq .Credentials.AccessKeyId |tr -d
 '"')
export AWS_SECRET_ACCESS_KEY=$(echo "$creds_json" |
 jq .Credentials.SecretAccessKey| tr -d '"')
export AWS_SESSION_TOKEN=$(echo "$creds_json" | jq .Credentials.SessionToken|tr -d
 '"')

export SERVICE_REGION=us-east-1 or us-east-2 or us-west-1 or us-west-2 or ca-
central-1 or
                      sa-east-1 or eu-north-1 or eu-west-1 or eu-west-2 or eu-
west-3 or eu-central-1 or me-south-1 or
                      me-central-1 or il-central-1 or af-south-1 or ap-east-1 or
 ap-northeast-1 or ap-northeast-2 or ap-southeast-1 or ap-southeast-2 or ap-south-1
 or
                      cn-north-1 or cn-northwest-1 or
                      us-gov-east-1 or us-gov-west-1
```

3. Conectar-se usando um dos seguintes métodos.

   - the section called “Console do Gremlin”
   - the section called “Gremlin Java”
   - the section called “SPARQL Java (RDF4J e Jena)”
   - the section called “Exemplo Python”

Configurando o AWS Lambda para a autenticação do Neptune IAM

AWS Lambda inclui credenciais automaticamente sempre que a função Lambda é executada.

Primeiro, adicione uma relação de confiança que conceda permissão para assumir o perfil ao serviço do Lambda.

Adicione o seguinte relacionamento de confiança ao perfil de autenticação do IAM do Neptune. Se você não tiver um perfil de autenticação do IAM do Neptune, consulte the section called “Tipos de política do IAM”.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Para obter informações sobre como ampliar a relação de confiança à função, consulte Editar a relação de confiança de uma função existente no Guia de administração do AWS Directory Service.

Se a política do Neptune ainda não estiver associada a um perfil, crie um perfil. Associe a política de autenticação do IAM do Neptune e, depois, adicione a política de confiança. Para obter informações sobre como criar uma função, consulte Criar uma função no Guia de administração do AWS Directory Service .

Como acessar o Neptune pelo Lambda

1. Faça login no AWS Management Console e abra o AWS Lambda console em https://console.aws.amazon.com/lambda/.
2. Crie uma nova função do Lambda para Python versão 3.6.
3. Atribua a função `AWSLambdaVPCAccessExecutionRole` para a função Lambda. Isso é necessário para acessar os recursos do Neptune, que são apenas VPC.
4. Atribua o perfil do IAM de autenticação do Neptune para a função do Lambda.

   Para obter mais informações, consulte AWS Lambda Permissions no Guia do desenvolvedor do AWS Lambda .
5. Copie o exemplo de Python de autenticação do IAM no código de função Lambda.

   Para obter mais informações sobre os exemplos e os códigos de exemplo, consulte the section called “Exemplo Python”.

Configurar o Amazon EC2 para autenticação do IAM do Neptune

O Amazon EC2 permite que você use perfis de instância para fornecer automaticamente as credenciais. Para obter mais informações, consulte Using Instance Profiles no Guia do usuário do IAM.

Primeiro, adicione uma relação de confiança que conceda permissão para assumir o perfil ao serviço do Amazon EC2.

Adicione o seguinte relacionamento de confiança ao perfil de autenticação do IAM do Neptune. Se você não tiver um perfil de autenticação do IAM do Neptune, consulte the section called “Tipos de política do IAM”.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Para obter informações sobre como ampliar a relação de confiança à função, consulte Editar a relação de confiança de uma função existente no Guia de administração do AWS Directory Service .

Se a política do Neptune ainda não estiver associada a um perfil, crie um perfil. Associe a política de autenticação do IAM do Neptune e, depois, adicione a política de confiança. Para obter informações sobre como criar uma função, consulte Criar uma função no Guia de administração do AWS Directory Service .

Para usar um script para obter as credenciais

1. Execute o seguinte comando para instalar comando jq. O script usa esse comando para analisar a saída do comando curl.

   ```
   sudo yum -y install jq
   ```

2. Crie um arquivo chamado `credentials.sh` em um editor de texto e adicione o texto a seguir. Substitua a região de serviço com o seu próprio valor.

```
role_name=$( curl -s http://169.254.169.254/latest/meta-data/iam/security-
credentials/ )
creds_json=$(curl -s http://169.254.169.254/latest/meta-data/iam/security-
credentials/${role_name})

export AWS_ACCESS_KEY_ID=$(echo "$creds_json" | jq .AccessKeyId |tr -d '"')
export AWS_SECRET_ACCESS_KEY=$(echo "$creds_json" | jq .SecretAccessKey| tr -d '"')
export AWS_SESSION_TOKEN=$(echo "$creds_json" | jq .Token|tr -d '"')

export SERVICE_REGION=us-east-1 or us-east-2 or us-west-1 or us-west-2 or ca-
central-1 or
                      sa-east-1 or eu-north-1 or eu-west-1 or eu-west-2 or eu-
west-3 or eu-central-1 or me-south-1 or
                      me-central-1 or il-central-1 or af-south-1 or ap-east-1 or
 ap-northeast-1 or ap-northeast-2 or ap-southeast-1 or ap-southeast-2 or ap-south-1
 or
                      cn-north-1 or cn-northwest-1 or
                      us-gov-east-1 or us-gov-west-1
```

3. Execute o script no shell `bash` usando o comando `source`:

```
source credentials.sh
```

   Melhor ainda é adicionar os comandos deste script ao arquivo `.bashrc` em sua instância do EC2 para que eles sejam invocados automaticamente quando você fizer login, disponibilizando credenciais temporárias para o console Gremlin.

4. Conectar-se usando um dos seguintes métodos.

   - the section called “Console do Gremlin”
   - the section called “Gremlin Java”
   - the section called “SPARQL Java (RDF4J e Jena)”
   - the section called “Exemplo Python”

Registro em log e monitoramento de recursos do Amazon Neptune

O Amazon Neptune é compatível com vários métodos para monitorar o desempenho e o uso:

- Status do cluster: confira a integridade do mecanismo de banco de dados do grafo de um cluster do Neptune. Para ter mais informações, consulte the section called “Status de instância”.
- Amazon CloudWatch — Neptune envia métricas automaticamente e também oferece suporte CloudWatch a alarmes. CloudWatch Para ter mais informações, consulte the section called “Usando CloudWatch”.
- Arquivos de log de auditoria: visualize, baixe ou observe arquivos de log de banco de dados usando o console do Neptune. Para ter mais informações, consulte the section called “Logs de auditoria com Neptune”.
- Publicação de registros no Amazon CloudWatch Logs — Você pode configurar um cluster de banco de dados Neptune para publicar dados de log de auditoria em um grupo de logs no Amazon Logs. CloudWatch Com o CloudWatch Logs, você pode realizar análises em tempo real dos dados de log, usar CloudWatch para criar alarmes e visualizar métricas, e usar o CloudWatch Logs para armazenar seus registros de log em um armazenamento altamente durável. Para ter mais informações, consulte Registros de Netuno CloudWatch .
- AWS CloudTrail— O Neptune suporta o registro de API usando. CloudTrail Para ter mais informações, consulte the section called “Registrando chamadas da API Neptune com AWS CloudTrail”.
- Marcação: use tags para adicionar metadados aos recursos do Neptune e monitorar o uso com base em tags. Para ter mais informações, consulte the section called “Marcar os recursos do Neptune”.

Validação de conformidade para o Amazon Neptune

Para saber se um AWS service (Serviço da AWS) está dentro do escopo de programas de conformidade específicos, consulte Serviços da AWS Escopo por Programa de Conformidade Serviços da AWS e escolha o programa de conformidade em que você está interessado. Para obter informações gerais, consulte Programas de AWS conformidade Programas AWS de .

Você pode baixar relatórios de auditoria de terceiros usando AWS Artifact. Para obter mais informações, consulte Baixar relatórios em AWS Artifact .

Sua responsabilidade de conformidade ao usar Serviços da AWS é determinada pela confidencialidade de seus dados, pelos objetivos de conformidade de sua empresa e pelas leis e regulamentações aplicáveis. AWS fornece os seguintes recursos para ajudar na conformidade:

- Guias de início rápido sobre segurança e conformidade — Esses guias de implantação discutem considerações arquitetônicas e fornecem etapas para a implantação de ambientes básicos AWS focados em segurança e conformidade.
- Arquitetura para segurança e conformidade com a HIPAA na Amazon Web Services — Este whitepaper descreve como as empresas podem usar AWS para criar aplicativos qualificados para a HIPAA.

> **Note**
>
> Nem todos Serviços da AWS são elegíveis para a HIPAA. Para mais informações, consulte a Referência dos serviços qualificados pela HIPAA.

- AWS Recursos de https://aws.amazon.com/compliance/resources/ de conformidade — Essa coleção de pastas de trabalho e guias pode ser aplicada ao seu setor e local.
- AWS Guias de conformidade do cliente — Entenda o modelo de responsabilidade compartilhada sob a ótica da conformidade. Os guias resumem as melhores práticas de proteção Serviços da AWS e mapeiam as diretrizes para controles de segurança em várias estruturas (incluindo o Instituto Nacional de Padrões e Tecnologia (NIST), o Conselho de Padrões de Segurança do Setor de Cartões de Pagamento (PCI) e a Organização Internacional de Padronização (ISO)).
- Avaliação de recursos com regras no Guia do AWS Config desenvolvedor — O AWS Config serviço avalia o quão bem suas configurações de recursos estão em conformidade com as práticas internas, as diretrizes e os regulamentos do setor.
- AWS Security Hub— Isso AWS service (Serviço da AWS) fornece uma visão abrangente do seu estado de segurança interno AWS. O Security Hub usa controles de segurança para avaliar os recursos da AWS e verificar a conformidade com os padrões e as práticas recomendadas do setor de segurança. Para obter uma lista dos serviços e controles aceitos, consulte a Referência de controles do Security Hub.
- Amazon GuardDuty — Isso AWS service (Serviço da AWS) detecta possíveis ameaças às suas cargas de trabalho Contas da AWS, contêineres e dados monitorando seu ambiente em busca de atividades suspeitas e maliciosas. GuardDuty pode ajudá-lo a atender a vários requisitos de conformidade, como o PCI DSS, atendendo aos requisitos de detecção de intrusões exigidos por determinadas estruturas de conformidade.
- AWS Audit Manager— Isso AWS service (Serviço da AWS) ajuda você a auditar continuamente seu AWS uso para simplificar a forma como você gerencia o risco e a conformidade com as regulamentações e os padrões do setor.

Resiliência no Amazon Neptune

A infraestrutura AWS global é construída em torno de AWS regiões e zonas de disponibilidade. AWS As regiões fornecem várias zonas de disponibilidade fisicamente separadas e isoladas, conectadas a redes de baixa latência, alta taxa de transferência e alta redundância. Com as Zonas de Disponibilidade, é possível projetar e operar aplicações e bancos de dados que executem o failover automaticamente entre as Zonas de Disponibilidade sem interrupção. As zonas de disponibilidade são mais altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de data center tradicionais.

Um cluster de banco de dados do Amazon Neptune só pode ser criado em uma Amazon VPC com pelo menos duas sub-redes em pelo menos duas zonas de disponibilidade. Ao distribuir as instâncias de cluster em pelo menos duas zonas de disponibilidade, o Neptune ajuda a garantir que haverá instâncias disponíveis no cluster de banco de dados no caso improvável de ocorrer uma falha na zona de disponibilidade. O volume do cluster de banco de dados do Neptune sempre abrange três zonas de disponibilidade para fornecer um armazenamento durável com menor possibilidade de perda de dados.

Para obter mais informações sobre AWS regiões e zonas de disponibilidade, consulte Infraestrutura AWS global.

Migrar um grafo existente para o Amazon Neptune

Há várias ferramentas e técnicas que podem ajudar você a migrar dados grafos existentes de outro datastore para o Amazon Neptune.

Um fluxo de trabalho de migração simples envolve as seguintes etapas:

- Exportar os dados do armazenamento existente para o Amazon Simple Storage Service (Amazon S3).
- Limpá-los e formatá-los para importação.
- Carregá–los em um cluster de banco de dados do Neptune usando o Carregador em massa do Neptune.
- Configurar a aplicação Gremlin ou SPARQL para usar o endpoint correspondente fornecido pelo Neptune.

> **Note**
>
> O cluster do Neptune deve estar sendo executado em uma VPC acessível à aplicação.

Há maneiras de simplificar e automatizar algumas dessas etapas, dependendo de onde os dados estão armazenados:

Tópicos

Migrar do Neo4j para o Amazon Neptune

O Neo4j e o Amazon Neptune são bancos de dados de grafos projetados para workloads de grafos transacionais on-line que são compatíveis com o modelo de dados de grafos de propriedades rotulado. Essas semelhanças tornam o Neptune uma escolha comum para clientes que desejam migrar as aplicações Neo4j atuais. No entanto, essas migrações não são simples procedimentos de mover sem alterações (lift-and-shift), pois há diferenças no suporte a linguagens e atributos, nas características operacionais, na arquitetura do servidor e nos recursos de armazenamento entre os dois bancos de dados.

Esta página explica o processo de migração e aborda aspectos a serem considerados antes de migrar uma aplicação de grafos Neo4j para o Neptune. Essas considerações geralmente se aplicam a qualquer aplicação de grafos Neo4j, seja alimentada por um banco de dados Community, Enterprise ou Aura. Embora cada solução seja exclusiva e possa exigir procedimentos adicionais, todas as migrações seguem o mesmo padrão geral.

Cada uma das etapas descritas nas seções a seguir inclui considerações e recomendações para simplificar o processo de migração. Além disso, há ferramentas de código aberto e postagens no blog que descrevem o processo e uma seção sobre compatibilidade de atributos com opções de arquitetura recomendadas.

Tópicos

Informações gerais sobre a migração do Neo4j para o Neptune

Com o suporte para a linguagem de consulta openCypher, você pode mover a maioria das workloads do Neo4j que usam o protocolo Bolt ou HTTPS para o Neptune. No entanto, o openCypher é

uma especificação de código aberto que contém a maioria, mas não todas, as funcionalidades compatíveis com outros bancos de dados, como o Neo4j.

Apesar de ser compatível de várias maneiras, o Neptune não é um substituto imediato do Neo4j. O Neptune é um serviço de banco de dados de grafos totalmente gerenciado com atributos corporativos, como alta disponibilidade e alta durabilidade, que é arquitetonicamente diferente do Neo4j. O Neptune é baseado em instâncias, com uma única instância de gravador principal e até 15 instâncias de réplica de leitura que permitem escalar a capacidade de leitura horizontalmente. Usando o Neptune Serverless, é possível aumentar ou reduzir a escala da capacidade computacional verticalmente, dependendo do volume de consultas. Isso é independente do armazenamento do Neptune, que é escalado automaticamente à medida que você adiciona dados.

O Neptune é compatível com a especificação padrão do openCypher de código aberto, versão 9. Na AWS, acreditamos que o código aberto é bom para todos e estamos comprometidos em trazer o valor do código aberto aos nossos clientes e a excelência operacional da AWS para as comunidades de código aberto.

No entanto, muitas aplicações executadas no Neo4j também usam atributos exclusivos que não são de código aberto e que o Neptune não aceita. Por exemplo, o Neptune não é compatível com procedimentos APOC, algumas cláusulas e funções específicas do Cypher e os tipos de dados `Char`, `Date` e `Duration`. O Neptune converte automaticamente os tipos de dados ausentes em tipos de dados compatíveis.

Além do openCypher, o Neptune também aceita a linguagem de consulta Apache TinkerPop Gremlin para grafos de propriedades (bem como SPARQL para dados do RDF). O Gremlin pode interoperar com o openCypher no mesmo grafo de propriedades e, em muitos casos, é possível usar o Gremlin para fornecer funcionalidades não oferecidas pelo openCypher. Veja uma comparação rápida das duas linguagens:

	openCypher	Gremlin
Style (Estilo)	Declarativa	Imperativa

Sintaxe	Correspondência de padrão
	`Match p=(a)-[:route]->(d)`
	`WHERE a.code='ANC'`
		Baseada em percurso
		`g.V().has('code', 'ANC').`
		`out('route').path().`

	openCypher	Gremlin
	`RETURN p`	`by(elementMap())`
Facilidade de uso	Inspirada em SQL, legível por não programadores	Curva de aprendizado mais íngreme, semelhante a linguagens de programação como Java
Flexibilidade	Baixo	Alto
Compatível com consultas	Consultas baseadas em strings	Consultas baseadas em strings ou código em linha compatíveis com bibliotecas de cliente
Clientes	HTTPS e Bolt	HTTPS e Websockets

Em geral, não é necessário alterar o modelo de dados para migrar do Neo4j para o Neptune, porque tanto o Neo4j quanto o Neptune são compatíveis com dados de grafos de propriedades rotulados (LPG). No entanto, o Neptune tem algumas diferenças de arquitetura e modelo de dados que você pode aproveitar para otimizar o desempenho. Por exemplo:

- Os IDs do Neptune são tratados como cidadãos de primeira classe.
- O Neptune usa políticas do AWS Identity and Access Management (IAM) para proteger o acesso aos dados de grafos de forma flexível e granular.
- O Neptune oferece várias maneiras de usar os cadernos Jupyter para executar consultas e visualizar os resultados. O Neptune também trabalha com ferramentas de visualização de terceiros.
- Embora o Neptune não tenha um substituto imediato para a biblioteca Neo4j Graph Data Science (GDS), no momento, ele é compatível com a análise de grafos por meio de uma variedade de soluções. Por exemplo, vários exemplos de blocos de anotações demonstram como aproveitar a integração do Neptune com o SDK do AWS Pandas em ambientes Python para executar análises em dados de grafos.

Em caso de dúvidas, entre em contato com o AWS Support ou com a equipe da conta da AWS. Usamos seus comentários para priorizar novos atributos que atendam às suas necessidades.

Preparar-se para migrar do Neo4j para o Neptune

Abordagens da migração

Ao migrar uma aplicação Neo4j para o Neptune, recomendamos uma das duas estratégias: reformulação da plataforma ou refatoração/rearquitetura. Para obter mais informações sobre estratégias de migração, consulte 6 Strategies for Migrating Applications to the Cloud, uma postagem no blog de Stephen Orban.

A abordagem de reformulação da plataforma, às vezes chamada de lift-tinker-and-shift, envolve as seguintes etapas:

- Identifique os casos de uso que a aplicação pretende satisfazer.
- Modifique o modelo de dados de grafos e a arquitetura de aplicações existentes para melhor atender a essas necessidades de workload usando os recursos do Neptune.
- Determine como migrar dados, consultas e outras partes da aplicação de origem para o modelo e a arquitetura de destino.

Essa abordagem retroativa permite migrar a aplicação para o tipo de solução do Neptune que você pode criar se esse fosse um projeto totalmente novo.

A abordagem de refatoração, por outro lado, envolve:

- Identificar os componentes da implementação existente, incluindo infraestrutura, dados, consultas e recursos de aplicações.
- Encontrar equivalentes no Neptune que possam ser usados para criar uma implementação comparável.

Essa abordagem progressiva busca trocar uma implementação por outra.

Na prática, é provável que você adote uma combinação dessas duas abordagens. É possível começar com um caso de uso, projetar a arquitetura do Neptune de destino, mas depois recorrer à implementação existente do Neo4j para identificar restrições e invariantes que você precisará manter. Por exemplo, talvez você precise continuar a integração com outros sistemas externos ou continuar oferecendo APIs específicas aos consumidores da aplicação de grafos. Com essas informações, é possível determinar quais dados já existem para serem migrados para o modelo de destino e quais devem ser obtidos em outro lugar.

Em outros pontos, você pode começar analisando uma parte específica da implementação do Neo4j como a melhor fonte de informações sobre o trabalho que a aplicação se destina a realizar. Esse tipo de arqueologia na aplicação existente pode ajudar a definir um caso de uso que você pode depois projetar para usar os recursos do Neptune.

Se você estiver criando uma aplicação usando o Neptune ou migrando uma aplicação existente do Neo4j, recomendamos trabalhar retroativamente a partir dos casos de uso para criar um modelo de dados, um conjunto de consultas e uma arquitetura de aplicações que atenda às necessidades empresariais.

Diferenças arquitetônicas entre o Neptune e o Neo4j

Quando os clientes consideram pela primeira vez migrar uma aplicação do Neo4j para o Neptune, geralmente é tentador fazer uma comparação do tipo “igual para igual” com base no tamanho da instância. No entanto, as arquiteturas do Neo4j e do Neptune têm diferenças fundamentais. O Neo4j é baseado em uma abordagem completa em que o carregamento de dados, o ETL de dados, as consultas de aplicações, o armazenamento de dados e as operações de gerenciamento acontecem no mesmo conjunto de recursos computacionais, como instâncias do EC2.

O Neptune, por outro lado, é um banco de dados de grafos focado em OLTP em que a arquitetura separa as responsabilidades e onde os recursos são dissociados para que possam ser escalados de forma dinâmica e independente.

Ao migrar do Neo4j para o Neptune, determine os requisitos de durabilidade, disponibilidade e escalabilidade dos dados da aplicação. A arquitetura de cluster do Neptune simplifica o design de aplicações que exigem alta durabilidade, disponibilidade e escalabilidade. Com uma compreensão da arquitetura de cluster do Neptune, é possível então projetar uma topologia de cluster do Neptune para atender a esses requisitos.

Arquitetura de cluster do Neo4j

Muitas aplicações de produção usam o agrupamento causal do Neo4j para oferecer durabilidade de dados, alta disponibilidade e escalabilidade. A arquitetura de agrupamento do Neo4j usa instâncias de servidor central e de réplica de leitura:

- Os servidores centrais oferecem durabilidade de dados e tolerância a falhas replicando dados com o protocolo Raft.
- As réplicas de leitura usam o envio de logs de transações para replicar dados de forma assíncrona para workloads de alta taxa de throughput de leitura.

Cada instância em um cluster, seja servidor central ou réplica de leitura, contém uma cópia completa dos dados de grafos.

Arquitetura do cluster do Neptune

Um cluster do Neptune é composto por uma instância de gravador principal e até 15 instâncias de réplica de leitura. Todas as instâncias no cluster compartilham o mesmo serviço de armazenamento distribuído subjacente que é separado das instâncias.

- A instância de gravador principal coordena todas as operações de gravação no banco de dados e é escalável verticalmente para oferecer suporte flexível para diferentes workloads de gravação. Ela também é compatível com operações de leitura.
- As instâncias de réplica de leitura são compatíveis com operações de leitura do volume de armazenamento subjacente e permitem que você escale horizontalmente para suportar altas workloads de leitura. Elas também oferecem alta disponibilidade servindo como destinos de failover para a instância principal.

  > **Note**
  >
  > Para workloads de gravação pesadas, é melhor escalar as instâncias da réplica de leitura para o mesmo tamanho da instância de gravador, a fim de garantir que os leitores possam permanecer consistentes com as alterações dos dados.

- O volume de armazenamento subjacente escala a capacidade de armazenamento automaticamente à medida que os dados no banco de dados aumentam, até 128 tebibytes (TiB) de armazenamento.

Os tamanhos das instâncias são dinâmicos e independentes. Enquanto o cluster estiver em execução, todas as instâncias poderão ser redimensionadas e as réplicas de leitura poderão ser adicionadas ou removidas.

O atributo Neptune Serverless pode aumentar e reduzir verticalmente a escala da capacidade computacional de modo automático à medida que a demanda aumenta e diminui. Isso não só pode diminuir a sobrecarga administrativa, mas também permite configurar o banco de dados para lidar com grandes picos de demanda sem prejudicar o desempenho nem exigir provisionamento excessivo.

É possível interromper um cluster de banco de dados por até sete dias.

O Neptune também é compatível com o ajuste de escala automático, para ajustar automaticamente os tamanhos das instâncias de leitor com base na workload.

Usando o atributo de banco de dados global do Neptune, é possível espelhar um cluster em até cinco outras regiões.

O Neptune também é tolerante a falhas por design:

- O volume do cluster que oferece armazenamento de dados para todas as instâncias do cluster abrange várias zonas de disponibilidade (AZs) em uma única Região da AWS. Cada AZ contém uma cópia completa dos dados do cluster.
- Se a instância principal ficar indisponível, o Neptune automaticamente fará failover para uma réplica de leitura existente sem perda de dados, normalmente em menos de trinta segundos. Se não houver réplicas de leitura existentes no cluster, o Neptune provisionará automaticamente uma nova instância principal: novamente, sem perda de dados.

O que tudo isso significa é que, ao migrar de um cluster causal do Neo4j para o Neptune, você não precisa arquitetar a topologia do cluster explicitamente para obter alta durabilidade e alta disponibilidade dos dados. Isso permite dimensionar o cluster para as workloads de leitura e gravação esperadas e todos os requisitos de maior disponibilidade que você possa ter, de apenas algumas maneiras:

- Para escalar as operações de leitura, adicione instâncias de réplica de leitura ou habilite a funcionalidade Neptune Serverless.
- Para melhorar a disponibilidade, distribua a instância principal e as réplicas de leitura no cluster em várias zonas de disponibilidade (AZs).
- Para reduzir o tempo de failover, provisione pelo menos uma instância de réplica de leitura que possa servir como destino de failover para a principal. É possível determinar a ordem em que as instâncias de réplica de leitura são promovidas à instância principal após uma falha, atribuindo uma prioridade a cada réplica. É uma prática recomendada garantir que um destino de failover tenha uma classe de instância capaz de lidar com a workload de gravação da aplicação se for promovido para principal.

Diferenças de armazenamento de dados entre o Neptune e o Neo4j

O Neptune usa um modelo de dados de grafos baseado em um modelo quádruplo nativo. Ao migrar os dados para o Neptune, há algumas diferenças na arquitetura do modelo de dados e da

camada de armazenamento que você deve conhecer para fazer o melhor uso do armazenamento compartilhado distribuído e escalável que o Neptune oferece:

- O Neptune não usa nenhum esquema explicitamente definido nem restrições. Ele permite que você adicione nós, bordas e propriedades dinamicamente sem precisar definir o esquema com antecedência. O Neptune não limita os valores e os tipos de dados armazenados, exceto conforme indicado em Limites do Neptune. Como parte da arquitetura de armazenamento do Neptune, os dados também são indexados automaticamente de forma a lidar com muitos dos padrões de acesso mais comuns. Essa arquitetura de armazenamento elimina a sobrecarga operacional da criação e do gerenciamento do esquema do banco de dados e da otimização do índice.
- O Neptune oferece uma arquitetura exclusiva de armazenamento distribuído e compartilhado que é escalada automaticamente em blocos de 10 GB à medida que as necessidades de armazenamento do banco de dados aumentam, até 128 tebibytes (TiB). Essa camada de armazenamento é confiável, durável e tolerante a falhas, com dados copiados seis vezes, duas vezes em cada uma das três zonas de disponibilidade. Por padrão, ele oferece a todos os clusters do Neptune uma camada de armazenamento de dados altamente disponível e tolerante a falhas. A arquitetura de armazenamento do Neptune reduz os custos e elimina a necessidade de provisionar ou provisionar em excesso o armazenamento para lidar com o crescimento futuro dos dados.

Antes de migrar os dados para o Neptune, é bom conhecer o modelo de dados de grafos de propriedades e a semântica das transações do Neptune.

Diferenças operacionais entre o Neptune e o Neo4j

O Neptune é um serviço totalmente gerenciado que automatiza muitas das tarefas operacionais normais que você precisaria realizar ao usar bancos de dados on-premises ou autogerenciados, como Neo4j Enterprise ou Community Edition:

- Backups automatizados: o Neptune faz backup automático do volume do cluster e retém o backup por um período de retenção especificado por você (de um a 35 dias). Esses backups são contínuos e incrementais para que você possa restaurar rapidamente em qualquer ponto do período de retenção. Quando os dados do backup estão sendo gravados, não há nenhum impacto sobre a performance ou interrupção de serviço do banco de dados.
- Snapshots manuais: o Neptune permite que você faça um snapshot do volume de armazenamento do cluster de banco de dados para fazer backup de todo o cluster de banco de dados. Esse tipo de snapshot pode então ser usado para restaurar o banco de dados, fazer uma cópia dele e compartilhá-lo entre contas.

- Clonagem: o Neptune é compatível com um atributo de clonagem que permite criar clones econômicos de um banco de dados com rapidez. Os clones usam um protocolo “copy-on-write” para exigir apenas um mínimo de espaço adicional depois de criados. A clonagem de banco de dados é uma forma eficaz de testar novos atributos ou atualizações do Neptune sem interromper o cluster de origem.
- Monitoramento: o Neptune oferece vários métodos para monitorar o desempenho e o uso do cluster, incluindo:
  - Status da instância
  - Integração com o Amazon CloudWatch e AWS CloudTrail
  - Recursos de log de auditoria
  - Notificações de eventos
  - Marcação
- Segurança: o Neptune oferece um ambiente seguro por padrão. Um cluster reside em uma VPC privada que oferece isolamento de rede de outros recursos. Todo o tráfego é criptografado via SSL e todos os dados são criptografados em repouso usando AWS KMS.

  Além disso, o Neptune se integra ao AWS Identity and Access Management (IAM) para oferecer autenticação. Ao especificar as chaves de condição do IAM, é possível usar as políticas do IAM para oferecer controle de acesso refinado sobre as ações de dados.

Diferenças de ferramentas e integração entre o Neptune e o Neo4j

O Neptune tem uma arquitetura para integrações e ferramentas diferente do Neo4j, o que pode afetar a arquitetura da aplicação. O Neptune usa os recursos computacionais do cluster para processar consultas, mas aproveita outros serviços da AWS de primeira classe para funcionalidades como pesquisa de texto completo (usando OpenSearch), ETL (usando Glue), etc. Para obter uma lista completa dessas integrações, consulte Integrações do Neptune.

Provisionar infraestrutura ao migrar do Neo4j para o Neptune

Os clusters do Amazon Neptune são criados para serem escalados em três dimensões: armazenamento, capacidade de gravação e capacidade de leitura. As seções abaixo abordam opções específicas a serem consideradas ao migrar.

Provisionar armazenamento

O armazenamento de qualquer cluster do Neptune é provisionado automaticamente, sem nenhuma sobrecarga administrativa de sua parte. Ele é redimensionado dinamicamente em blocos de 10 GB à medida que as necessidades de armazenamento do cluster aumentam. Como resultado, não há necessidade de estimar e provisionar ou provisionar em excesso o armazenamento para lidar com o crescimento futuro dos dados.

Provisionar a capacidade de gravação

O Neptune oferece uma única instância de gravador que pode ser escalada verticalmente para qualquer tamanho de instância disponível na página Preços do Neptune. Ao ler e gravar dados em uma instância de gravador, todas as transações são compatíveis com ACID, com isolamento de dados conforme definido em Níveis de isolamento de transação no Neptune.

A escolha de um tamanho ideal de uma instância de gravador exige a execução de testes de carga para determinar o tamanho ideal da instância para a workload. Qualquer instância no Neptune pode ser redimensionada a qualquer momento modificando a classe da instância de banco de dados. É possível estimar o tamanho da instância inicial com base na simultaneidade e na latência média da consulta, conforme descrito abaixo em Estimar o tamanho ideal da instância ao provisionar o cluster.

Provisionar a capacidade de leitura

O Neptune foi criado para escalar instâncias de réplica de leitura horizontalmente, adicionando até 15 delas a um cluster (ou mais em um banco de dados global do Neptune) e verticalmente a qualquer tamanho de instância disponível na página Preços do Neptune. Todas as instâncias de réplica de leitura do Neptune usam o mesmo volume de armazenamento subjacente, permitindo a replicação transparente dos dados com o mínimo de atraso.

Além de permitir a escalabilidade horizontal das solicitações de leitura em um cluster do Neptune, as réplicas de leitura também atuam como destinos de failover para a instância de gravador a fim de permitir alta disponibilidade. Consulte Diretrizes operacionais básicas do Amazon Neptune para obter sugestões sobre como determinar o número e o posicionamento apropriados das réplicas de leitura no cluster.

Para aplicações em que a conectividade e a workload sejam imprevisíveis, o Neptune também é compatível com um atributo de ajuste de escala automático que pode ajustar automaticamente o número de réplicas do Neptune com base nos critérios especificados.

Para determinar o tamanho e o número ideais das instâncias de réplica de leitura, é necessário realizar testes de carga para determinar as características da workload de leitura que elas devem aceitar. Qualquer instância no Neptune pode ser redimensionada a qualquer momento modificando a classe da instância de banco de dados. É possível estimar o tamanho da instância inicial com base na simultaneidade e na latência média da consulta, conforme descrito na próxima seção.

Use o Neptune Serverless para escalar instâncias de leitura e gravação automaticamente, conforme necessário

Embora muitas vezes seja útil poder estimar a capacidade computacional que as workloads previstas exigirão, você pode configurar o atributo Neptune Serverless para aumentar e reduzir verticalmente a escala da capacidade de leitura e gravação de modo automático. Isso pode ajudar você a atender aos requisitos de pico e, ao mesmo tempo, reduzir automaticamente quando a demanda diminui.

Estimar o tamanho ideal da instância ao provisionar o cluster

A estimativa do tamanho ideal da instância exige conhecer a latência média da consulta no Neptune, quando a workload está em execução, bem como o número de consultas simultâneas processadas. Uma estimativa aproximada do tamanho da instância pode ser calculada como a latência média da consulta multiplicada pelo número de consultas simultâneas. Isso fornece o número médio de threads simultâneos necessários para lidar com a workload.

Cada vCPU em uma instância do Neptune pode aceitar dois threads de consulta simultâneos, portanto, dividir os threads por dois fornece o número de vCPUs necessárias, que podem ser correlacionadas ao tamanho apropriado da instância na página Preços do Neptune. Por exemplo:

```
Average Query Latency:         30ms (0.03s)
Number of concurrent queries:  1000/second

Number of threads needed:      0.03 x 1000 = 30 threads
Number of vCPUs needed:        30 / 2 = 15 vCPUs
```

Correlacionando isso com o número de vCPUs em uma instância, vemos que obtemos uma estimativa aproximada de que a `r5.4xlarge` seria a instância recomendada para testar essa workload. Essa estimativa é aproximada e serve apenas para oferecer orientação inicial sobre

a seleção do tamanho da instância. Qualquer aplicação deve passar por um exercício de dimensionamento correto para determinar o número e os tipos apropriados de instâncias adequadas à workload.

Os requisitos de memória também devem ser levados em consideração, assim como os requisitos de processamento. O Neptune tem melhor desempenho quando os dados acessados pelas consultas estão disponíveis no cache do grupo de buffer da memória principal. O provisionamento de memória suficiente também pode reduzir significativamente os custos de E/S.

Mais detalhes e orientações sobre o dimensionamento de instâncias em um cluster do Neptune podem ser encontrados na página Dimensionar instâncias de banco de dados em um cluster de banco de dados do Neptune.

Migração de dados do Neo4j para o Neptune

Ao realizar uma migração do Neo4j para o Amazon Neptune, a migração dos dados é uma etapa importante do processo. Há várias abordagens para migrar dados. A abordagem correta é determinada pelas necessidades da aplicação, pelo tamanho dos dados e pelo tipo de migração desejada. No entanto, muitas dessas migrações exigem a avaliação das mesmas considerações, das quais várias são destacadas abaixo.

> **Note**
>
> Consulte Migrating a Neo4j graph database to Neptune with a fully automated utility no blog do banco de dados da AWS para obter uma explicação passo a passo completa de um exemplo de como realizar uma migração de dados off-line.

Avaliar a migração de dados do Neo4j para o Neptune

A primeira etapa ao avaliar qualquer migração de dados é determinar como você migrará os dados. As opções dependem da arquitetura da aplicação que está sendo migrada, do tamanho dos dados e das necessidades de disponibilidade durante a migração. Em geral, as migrações tendem a se enquadrar em uma das duas categorias: on-line ou off-line.

As migrações off-line tendem a ser as mais simples de realizar, porque a aplicação não aceita tráfego de leitura ou gravação durante a migração. Depois que a aplicação parar de aceitar tráfego, os dados poderão ser exportados, otimizados, importados e a aplicação testada antes de ser reabilitada.

As migrações on-line são mais complexas, pois a aplicação ainda precisa aceitar tráfego de leitura e gravação enquanto os dados estão sendo migrados. As necessidades exatas de cada migração on-line podem ser diferentes, mas geralmente a arquitetura seria semelhante à seguinte:

- Um feed de alterações contínuas no banco de dados precisa ser habilitado no Neo4j configurando os fluxos do Neo4j como fonte para um cluster do Kafka.
- Depois que isso for concluído, uma exportação do sistema em execução poderá ser feita, seguindo as instruções em Exportar dados do Neo4j ao migrar para o Neptune e o horário indicado para correlação posterior com o tópico do Kafka.
- Os dados exportados são então importados para o Neptune, seguindo as instruções em Importar dados do Neo4j ao migrar para o Neptune.

- Os dados alterados dos fluxos do Kafka podem então ser copiados no cluster do Neptune usando uma arquitetura semelhante à descrita em Writing to Amazon Neptune from Amazon Kinesis Data Streams. Observe que a replicação das alterações pode ser executada paralelamente para validar a arquitetura e o desempenho da nova aplicação.
- Depois que a migração de dados for validada, o tráfego da aplicação poderá ser redirecionado para o cluster do Neptune e a instância do Neo4j poderá ser desativada.

Otimizações de modelos de dados para migrar do Neo4j para o Neptune

Tanto o Neptune quanto o Neo4j são compatíveis com grafos de propriedades rotulados (LPG). No entanto, o Neptune tem algumas diferenças de arquitetura e modelo de dados que você pode aproveitar para otimizar o desempenho:

Otimizar IDs de nós e bordas

O Neo4j gera automaticamente IDs numéricos longos. Usando o Cypher, é possível se referir aos nós por ID, mas isso geralmente é desencorajado priorizando-se a pesquisa de nós por uma propriedade indexada.

O Neptune permite fornecer os próprios IDs baseados em strings para vértices e bordas. Se você não fornecer os próprios IDs, o Neptune vai gerar automaticamente representações de string de UUIDs para novas bordas e vértices.

Se você migrar dados do Neo4j para o Neptune exportando do Neo4j e depois importando em massa para o Neptune, poderá preservar os IDs do Neo4j. Os valores numéricos gerados pelo Neo4j podem atuar como IDs fornecidos pelo usuário ao importar para o Neptune, no qual são representados como strings em vez de valores numéricos.

No entanto, há circunstâncias em que convém promover uma propriedade de vértice para se tornar um ID de vértice. Assim como pesquisar um nó usando uma propriedade indexada é a maneira mais rápida de encontrar um nó no Neo4j, pesquisar um vértice por ID é a maneira mais rápida de encontrar um vértice no Neptune. Portanto, se você puder identificar uma propriedade de vértice adequada que contenha valores exclusivos, pense em substituir o ~id do vértice pelo valor da propriedade indicada nos arquivos CSV de carregamento em massa. Se você fizer isso, também precisará reescrever todos os valores correspondentes de borda ~from e ~to nos arquivos CSV.

Restrições do esquema ao migrar dados do Neo4j para o Neptune

No Neptune, a única restrição do esquema disponível é a exclusividade do ID de um nó ou uma borda. As aplicações que precisam aproveitar uma restrição de exclusividade são incentivadas a

analisar essa abordagem para obter uma restrição de exclusividade por meio da especificação do ID do nó ou da borda. Se a aplicação usou várias colunas como restrição de exclusividade, o ID pode ser definido como uma combinação desses valores. Por exemplo, `id=123, code='SEA'` pode ser representado como `ID='123_SEA'` para obter uma restrição de exclusividade complexa.

Otimização da direção da borda ao migrar dados do Neo4j para o Neptune

Quando nós, bordas ou propriedades são adicionados ao Neptune, eles são automaticamente indexados de três maneiras diferentes, com um quarto índice opcional. Devido à forma como o Neptune cria e usa os índices, as consultas que seguem as bordas de saída são mais eficientes do que as que usam bordas de entrada. Em termos do modelo de armazenamento de dados de grafos do Neptune, essas são pesquisas baseadas em assuntos que usam o índice SPOG.

Se, ao migrar o modelo de dados e consultas para o Neptune, você descobrir que as consultas mais importantes dependem de percorrer bordas de entrada onde há um alto grau de fanout, pense em alterar o modelo para que esses percursos sigam as bordas de saída, principalmente quando você não pode especificar quais rótulos de borda percorrer. Para fazer isso, inverta a direção das bordas relevantes e atualize os rótulos das bordas para refletir a semântica dessa mudança de direção. Por exemplo, você pode alterar:

```
person_A — parent_of — person_B
   to:
person_B — child_of — person_A
```

Para fazer essa alteração em um arquivo CSV de borda de carregamento em massa, basta trocar os cabeçalhos das colunas `~from` e `~to` e atualizar os valores da coluna `~label`.

Como alternativa à inversão da direção das bordas, você pode habilitar um quarto índice do Neptune, o índice OSGP, que torna o percurso das bordas de entrada ou as pesquisas baseadas em objetos muito mais eficiente. No entanto, habilitar esse quarto índice reduzirá as taxas de inserção e exigirá mais armazenamento.

Otimização da filtragem ao migrar dados do Neo4j para o Neptune

O Neptune é otimizado para funcionar melhor quando as propriedades são filtradas para a propriedade mais seletiva disponível. Quando vários filtros são usados, o conjunto de itens correspondentes é encontrado para cada um e, depois, a sobreposição de todos os conjuntos correspondentes é calculada. Quando possível, combinar várias propriedades em uma única propriedade minimiza o número de pesquisas de índice e diminui a latência de uma consulta.

Por exemplo, essa consulta usa duas pesquisas de índice e uma junção:

```
MATCH (n) WHERE n.first_name='John' AND n.last_name='Doe' RETURN n
```

Essa consulta recupera as mesmas informações usando uma única pesquisa de índice:

```
MATCH (n) WHERE n.name='John Doe' RETURN n
```

O Neptune é compatível com tipos de dados diferentes dos do Neo4j.

Mapeamentos de tipo de dados do Neo4j em tipos de dados compatíveis com o Neptune

- Lógico: `Boolean`

  Associe no Neptune a `Bool` ou a `Boolean`.
- Numérico: `Number`

  Associe no Neptune ao mais restrito dos seguintes tipos do openCypher no Neptune que pode aceitar todos os valores da propriedade numérica em questão:

  ```
  Byte
  Short
  Integer
  Long
  Float
  Double
  ```
- Texto: `String`

  Associe no Neptune a `String`.
- Ponto no tempo:

  ```
  Date
  Time
  LocalTime
  DateTime
  LocalDateTime
  ```

  Associe no Neptune a `Date` como UTC, usando um dos seguintes formatos ISO-8601 aceitos pelo Neptune:

```
yyyy-MM-dd
yyyy-MM-ddTHH:mm
yyyy-MM-ddTHH:mm:ss
yyyy-MM-ddTHH:mm:ssZ
```

- Duração do tempo: `Duration`

  Associe no Neptune a um valor numérico para aritmética de datas, se necessário.

- Espacial: `Point`

  Associe no Neptune a valores numéricos de componentes, cada um dos quais se torna uma propriedade separada, ou expresse como um valor de string a ser interpretado pela aplicação cliente. Observe que a integração de pesquisa de texto completo do Neptune usando o OpenSearch permite indexar propriedades de geolocalização.

Migrar propriedades de vários valores do Neo4j ao Neptune

O Neo4j permite que listas homogêneas de tipos simples sejam armazenadas como propriedades dos nós e das bordas. Essas listas podem conter valores duplicados.

O Neptune, no entanto, permite apenas cardinalidade definida ou única para propriedades de vértice e cardinalidade única para propriedades de borda em dados de grafos de propriedades. Como resultado, não há migração direta de propriedades da lista de nós do Neo4j que contenham valores duplicados para as propriedades dos vértices de Neptune nem das propriedades da lista de relacionamentos do Neo4j para as propriedades da borda do Neptune.

Algumas estratégias possíveis para migrar propriedades de nós de vários valores do Neo4j com valores duplicados para o Neptune são as seguintes:

- Descarte os valores duplicados e converta a propriedade do nó do Neo4j de vários valores em uma propriedade de vértice do Neptune de cardinalidade definida. Observe que o conjunto do Neptune pode não refletir a ordem dos itens na propriedade de vários valores do Neo4j original.
- Converta a propriedade de nó do Neo4j de vários valores em uma representação de string de uma lista formatada em JSON em uma propriedade de string de vértice do Neptune.
- Extraia cada um dos valores de propriedade de vários valores em um vértice separado com uma propriedade de valor e conecte esses vértices ao vértice pai usando uma borda rotulada com o nome da propriedade.

Da mesma forma, estratégias possíveis para migrar propriedades de relacionamento de vários valores do Neo4j para o Neptune são as seguintes:

- Converta a propriedade de relacionamento do Neo4j de vários valores em uma representação de string de uma lista formatada em JSON em uma propriedade de string de borda do Neptune.
- Refatore o relacionamento do Neo4j em bordas do Neptune de entrada e de saída conectadas a um vértice intermediário. Extraia cada um dos valores de propriedade de vários valores em um vértice separado com uma propriedade de valor e conecte esses vértices a esse vértice intermediário usando uma borda rotulada com o nome da propriedade.

Observe que uma representação de string de uma lista formatada em JSON é opaca para a linguagem de consulta openCypher, embora o openCypher inclua um predicado `CONTAINS` que permite pesquisas simples dentro de valores de string.

Exportar dados do Neo4j ao migrar para o Neptune

Ao exportar dados do Neo4j, use os procedimentos APOC para exportar para CSV ou GraphML. Embora seja possível exportar para outros formatos, existem ferramentas de código aberto para converter dados CSV exportados do Neo4j no formato de carregamento em massa do Neptune e também ferramentas de código aberto para converter dados do GraphML exportados do Neo4j para o formato de carregamento em massa do Neptune.

Também é possível exportar dados diretamente para o Amazon S3 usando os vários procedimentos APOC. A exportação para um bucket do Amazon S3 está desabilitada por padrão, mas pode ser habilitada usando os procedimentos destacados em Exporting to Amazon S3 na documentação do APOC no Neo4j.

Importar dados do Neo4j ao migrar para o Neptune

É possível importar dados para o Neptune usando o carregador em massa do Neptune ou usando a lógica da aplicação em uma linguagem de consulta compatível, como openCypher.

O carregador em massa do Neptune é a abordagem preferencial para importar grandes quantidades de dados, pois oferece desempenho de importação otimizado caso você siga as práticas recomendadas. O carregador em massa é compatível com dois formatos CSV diferentes, para os quais os dados exportados do Neo4j podem ser convertidos usando os utilitários de código aberto mencionados acima na seção Exportar dados.

Você também pode usar o openCypher para importar dados com lógica personalizada para análise, transformação e importação. É possível enviar as consultas do openCypher por meio do endpoint HTTPS (o que é recomendado) ou usando o driver bolt.

Migração de aplicações do Neo4j para o Neptune

Depois de migrar os dados do Neo4j para o Neptune, a próxima etapa é migrar a aplicação em si. Assim como acontece com os dados, há várias abordagens para migrar a aplicação com base nas ferramentas utilizadas, nos requisitos, nas diferenças de arquitetura, etc. Nesse processo, em geral, é necessário pensar nos fatores descritos abaixo.

Migrar conexões ao transferir dados do Neo4j para o Neptune

Se, no momento, você não usa os drivers Bolt ou gostaria de usar uma alternativa, pode se conectar ao endpoint HTTPS, que concede acesso total aos dados exibidos.

Se você tiver uma aplicação que use o protocolo Bolt, poderá migrar essas conexões com o Neptune e permitir que as aplicações se conectem usando os mesmos drivers utilizados no Neo4j. Para se conectar ao Neptune, talvez seja necessário fazer uma ou mais das seguintes alterações na aplicação:

- O URL e a porta precisarão ser atualizados para usar os endpoints e a porta do cluster (o padrão é 8182).
- O Neptune exige que todas as conexões usem SSL, então é necessário especificar para cada conexão que ela está criptografada.
- O Neptune gerencia a autenticação por meio da atribuição de políticas e perfis do IAM. As políticas e os perfis do IAM oferecem um nível extremamente flexível de gerenciamento de usuários na aplicação, por isso é importante ler e entender as informações na Visão geral do IAM antes de configurar o cluster.
- As conexões de parafusos se comportam de maneira diferente de várias maneiras no Neptune e no Neo4j, conforme explicado em Comportamento da conexão do Bolt no Neptune.
- É possível encontrar mais informações e sugestões em Práticas recomendadas para o Neptune ao usar openCypher e Bolt.

Há exemplos de código de linguagens comumente usadas, como Java, Python, .NET e NodeJS, e de cenários de conexão, como o uso da autenticação do IAM, em Usar o protocolo Bolt para fazer consultas do openCypher ao Neptune.

Direcionar consultas para instâncias de cluster ao migrar do Neo4j para o Neptune

As aplicações cliente do Neo4j usam um driver de roteamento e especificam um modo de acesso para direcionar solicitações de leitura e gravação para um servidor apropriado em um cluster causal.

Ao migrar uma aplicação cliente para o Neptune, use os endpoints do Neptune para direcionar consultas com eficiência para uma instância apropriada no cluster:

- Todas as conexões com o Neptune devem usar `bolt://` em vez de `bolt+routing://` `neo4j://` ou o URL.
- O endpoint de cluster conecta-se à instância principal atual do cluster. Use o endpoint de cluster para direcionar solicitações de gravação para a principal.
- O endpoint de leitor distribui conexões entre instâncias de réplica de leitura no cluster. Se você tiver um cluster de instância única sem instâncias de réplica de leitura, o endpoint de leitor se conectará à instância principal, que é compatível com operações de gravação. Se o cluster contiver uma ou mais instâncias de réplica de leitura, o envio de uma solicitação de gravação para o endpoint de leitor gerará uma exceção.
- Cada instância no cluster também pode ter o próprio endpoint de instância. Use um endpoint de instância se a aplicação cliente precisar enviar uma solicitação a uma instância específica no cluster.

Para obter mais informações, consulte Considerações sobre endpoint do Neptune.

Consistência de dados no Neptune

Ao usar clusters causais do Neo4j, as réplicas de leitura acabam sendo consistentes com os servidores centrais, mas as aplicações cliente podem garantir a consistência causal usando o encadeamento causal. O encadeamento causal envolve a transmissão de marcadores entre transações, o que permite que uma aplicação cliente grave em um servidor central e depois leia a própria gravação em uma réplica de leitura.

No Neptune, as instâncias de réplica de leitura acabam sendo consistentes com o gravador, com um atraso na réplica geralmente inferior a cem milissegundos. No entanto, até que uma alteração seja replicada, as atualizações nas bordas e nos vértices existentes e as adições de novas bordas e vértices não ficam visíveis em uma instância de réplica. Portanto, se a aplicação precisar de consistência imediata no Neptune lendo cada gravação, use o endpoint de cluster para a operação de leitura após gravação. Esse é o único momento de usar o endpoint de cluster para operações de leitura. Em todas as outras circunstâncias, use o endpoint de leitor para leituras.

Migrar consultas do Neo4j para o Neptune

Embora o suporte para openCypher reduza drasticamente a quantidade de trabalho necessária para migrar consultas do Neo4j, ainda há algumas diferenças a serem avaliadas durante a migração:

- Conforme abordado em Otimizações do modelo de dados acima, pode haver modificações no modelo de dados que você precise fazer para criar um modelo de dados de grafos otimizado para o Neptune o que, por sua vez, exigirá alterações nas consultas e nos testes.
- O Neo4j oferece uma série de extensões de linguagem específicas do Cypher que não estão incluídas na especificação do openCypher implementada pelo Neptune. Dependendo do caso de uso e do atributo utilizado, pode haver soluções alternativas na linguagem openCypher, no uso da linguagem Gremlin ou por meio de outros mecanismos, conforme descrito em Reformular consultas do Cypher para serem executadas no openCypher no Neptune.
- As aplicações geralmente usam outros componentes de middleware para interagir com o banco de dados, em vez dos próprios drivers do Bolt. Confira Compatibilidade do Neptune com o Neo4j para ver se as ferramentas ou o middleware que você está usando são compatíveis.
- No caso de um failover, o driver do Bolt pode continuar se conectando à instância de gravador ou de leitor anterior porque o endpoint de cluster fornecido à conexão foi resolvido para um endereço IP. O tratamento adequado de erros na aplicação deve resolver isso, conforme descrito em Criar uma conexão após o failover.
- Quando as transações são canceladas devido a conflitos não resolvidos ou a tempos limite de espera de bloqueio, o Neptune responde com uma `ConcurrentModificationException`. Para obter mais informações, consulte Códigos de erro do mecanismo. Como melhor prática, os clientes sempre devem capturar e lidar com essas exceções.

  A `ConcurrentModificationException` ocorre ocasionalmente quando vários encadeamentos ou várias aplicações estão gravando no sistema simultaneamente. Devido aos níveis de isolamento de transações, esses conflitos às vezes podem ser inevitáveis.
- O Neptune é compatível com a execução de consultas do Gremlin e do openCypher nos mesmos dados. Isso significa que, em algumas situações, talvez seja necessário pensar no uso do Gremlin, com os recursos de consulta mais poderosos, para realizar algumas das funcionalidades das consultas.

Conforme abordado em Provisionar infraestrutura acima, cada aplicação deve passar por um exercício de dimensionamento correto para garantir que o número de instâncias, os tamanhos das instâncias e a topologia do cluster sejam otimizados para a workload específica da aplicação.

As considerações abordadas aqui para migrar a aplicação são as mais comuns, mas não se trata de uma lista completa. Cada aplicação é exclusiva. Em caso de mais dúvidas, entre em contato com o AWS Support ou entre em contato com a equipe de sua conta.

Migrar atributos e ferramentas específicos do Neo4j

O Neo4j tem uma série de atributos e complementos personalizados com funcionalidades nas quais a aplicação pode confiar. Ao avaliar a necessidade de migrar essa funcionalidade, geralmente é útil investigar se existe uma abordagem melhor na AWS para concretizar o mesmo objetivo. Considerando as diferenças de arquitetura entre o Neo4j e o Neptune, muitas vezes é possível encontrar alternativas eficazes que utilizem outros serviços ou integrações da AWS.

Consulte Compatibilidade do Neptune com o Neo4j para obter uma lista de atributos específicos do Neo4j e soluções alternativas sugeridas.

Compatibilidade do Neptune com o Neo4j

O Neo4j tem uma abordagem completa em que o carregamento e o ETL de dados, as consultas de aplicações, o armazenamento de dados e as operações de gerenciamento acontecem no mesmo conjunto de recursos computacionais, como instâncias do EC2. O Amazon Neptune é um banco de dados de grafos de especificações abertas concentrado em OLTP em que a arquitetura separa as operações e desconecta os recursos para que eles possam ser escalados dinamicamente.

Há uma série de atributos e ferramentas no Neo4j, incluindo ferramentas de terceiros, que não fazem parte da especificação do openCypher, são incompatíveis com o openCypher ou com a implementação do openCypher pelo Neptune. Veja alguns dos mais comuns listados.

Atributos específicos do Neo4j não presentes no Neptune

- **`LOAD CSV`**: o Neptune tem uma abordagem arquitetônica diferente do Neo4j para carregar dados. Para permitir uma melhor escalabilidade e otimização de custos, o Neptune implementa uma separação de preocupações em relação aos recursos e recomenda o uso de uma das integrações de serviços da AWS, como AWS Glue para executar os processos de ETL necessários para preparar dados em um formato compatível com o carregador em massa do Neptune.

  Outra opção é fazer a mesma coisa usando o código da aplicação executado em recursos computacionais da AWS, como instâncias do Amazon EC2, funções do Lambda, Amazon Elastic Container Service, trabalhos AWS Batch, etc. O código pode usar o endpoint HTTPS do Neptune ou o endpoint do Bolt.
- Controle de acesso refinado: o Neptune é compatível com o controle de acesso granular sobre ações de acesso a dados usando chaves de condição do IAM. Controle de acesso refinado adicional pode ser implementado na camada de aplicação.
- Neo4j Fabric: o Neptune é compatível com a federação de consultas em bancos de dados para workloads do RDF usando a palavra-chave `SERVICE` do SPARQL. Como no momento não há um padrão aberto nem uma especificação para federação de consultas para workloads de grafos de propriedades, essa funcionalidade precisaria ser implementada na camada da aplicação.
- Controle de acesso baseado em perfil (RBAC): o Neptune gerencia a autenticação por meio da atribuição de políticas e perfis do IAM. As políticas e os perfis do IAM oferecem um nível extremamente flexível de gerenciamento de usuários em um aplicação, por isso é útil ler e entender as informações na Visão geral do IAM antes de configurar o cluster.
- Marcação: os clusters do Neptune consistem em uma única instância de gravador e até 15 instâncias de réplica de leitura. Os dados gravados na instância de gravador são compatíveis

com ACID e fornecem uma forte garantia de consistência nas leituras subsequentes. As réplicas de leitura usam o mesmo volume de armazenamento da instância de gravador e acabam sendo consistentes, em geral, em menos de 100 ms a partir do momento em que os dados são gravados. Se o caso de uso tiver uma necessidade imediata de garantir a consistência de leitura de novas gravações, essas leituras deverão ser direcionadas para o endpoint de cluster em vez do endpoint de leitor.

- Procedimentos APOC: como os procedimentos APOC não estão incluídos na especificação do openCypher, o Neptune não oferece suporte direto para procedimentos externos. Em vez disso, o Neptune depende de integrações com outros serviços da AWS para obter uma funcionalidade semelhante ao usuário final de maneira escalável, segura e sólida. Às vezes, os procedimentos APOC podem ser regravados em openCypher ou em Gremlin, e alguns não são relevantes para as aplicações do Neptune.

  Em geral, os procedimentos APOC se enquadram nas categorias abaixo:

  - Importar: o Neptune é compatível com a importação de dados com uma variedade de formatos usando linguagens de consulta, o carregador em massa do Neptune ou como destino do AWS Database Migration Service. As operações de ETL nos dados podem ser realizadas com AWS Glue e o pacote de código aberto `neptune-python-utils`.
  - Exportar: o Neptune é compatível com a exportação de dados usando o utilitário `neptune-export`, que é compatível com uma variedade de formatos e métodos de exportação comuns.
  - Integração de banco de dados: o Neptune é compatível com a integração com outros bancos de dados usando ferramentas ETL, como AWS Glue ou ferramentas de migração, como o AWS Database Migration Service.
  - Atualizações de grafos: o Neptune é compatível com um rico conjunto de atributos para atualizar dados de grafos de propriedades por meio do suporte para as linguagens de consulta openCypher e Gremlin. Consulte Reformulações do Cypher para obter exemplos de regravações de procedimentos usados com frequência.
  - Estruturas de dados: o Neptune é compatível com um rico conjunto de atributos para atualizar dados de grafos de propriedades por meio do suporte para as linguagens de consulta openCypher e Gremlin. Consulte Reformulações do Cypher para obter exemplos de regravações de procedimentos usados com frequência.
  - Temporal (data e hora): o Neptune é compatível com um rico conjunto de atributos para atualizar dados de grafos de propriedades por meio do suporte para as linguagens de consulta openCypher e Gremlin. Consulte Reformulações do Cypher para obter exemplos de regravações de procedimentos usados com frequência.

  - Matemático: o Neptune é compatível com um rico conjunto de atributos para atualizar dados de grafos de propriedades por meio do suporte para as linguagens de consulta openCypher e Gremlin. Consulte Reformulações do Cypher para obter exemplos de regravações de procedimentos usados com frequência.
  - Consultas de grafos avançadas: o Neptune é compatível com um rico conjunto de atributos para atualizar dados de grafos de propriedades por meio do suporte para as linguagens de consulta openCypher e Gremlin. Consulte Reformulações do Cypher para obter exemplos de regravações de procedimentos usados com frequência.
  - Comparação de grafos: o Neptune é compatível com um rico conjunto de atributos para atualizar dados de grafos de propriedades por meio do suporte para as linguagens de consulta openCypher e Gremlin. Consulte Reformulações do Cypher para obter exemplos de regravações de procedimentos usados com frequência.
  - Execução do Cypher: o Neptune é compatível com um rico conjunto de atributos para atualizar dados de grafos de propriedades por meio do suporte para as linguagens de consulta openCypher e Gremlin. Consulte Reformulações do Cypher para obter exemplos de regravações de procedimentos usados com frequência.
- Procedimentos personalizados: o Neptune não é compatível com procedimentos personalizados criados por usuários. Essa funcionalidade teria que ser implementada na camada da aplicação.
- Geoespacial: embora o Neptune não ofereça suporte nativo para atributos geoespaciais, uma funcionalidade semelhante pode ser obtida por meio da integração com outros serviços da AWS, conforme mostrado nesta postagem no blog: Combine Amazon Neptune and Amazon OpenSearch Service for geospatial queries de Ross Gabay e Abhilash Vinod (1.º de fevereiro de 2022).
- Ciência de dados de grafos: o Neptune oferece suporte à análise de grafos por meio do Neptune Analytics, um mecanismo otimizado para memória compatível com uma biblioteca de algoritmos analíticos de grafos.

  O Neptune também oferece integração com o SDK do AWS Pandas e vários exemplos de bloco de anotações que mostram como aproveitar essa integração em ambientes Python para realizar análises em dados de grafos.

- Restrições do esquema: no Neptune, a única restrição do esquema disponível é a exclusividade do ID de um nó ou uma borda. Não há nenhum atributo para especificar restrições adicionais do esquema nem qualquer restrição adicional de exclusividade ou valor em um elemento no grafo. Os valores de ID no Neptune são strings e podem ser definidos usando Gremlin, da seguinte forma:

```
g.addV('person').property(id, '1') )
```

As aplicações que precisam utilizar o ID como restrição de exclusividade são incentivadas a tentar essa abordagem para obter uma restrição de exclusividade. Se a aplicação usou várias colunas como restrição de exclusividade, o ID pode ser definido como uma combinação desses valores. Por exemplo, `id=123, code='SEA'` pode ser representado como `ID='123_SEA'` para obter uma restrição de exclusividade complexa.

- Multilocação: o Neptune é compatível apenas com um único grafo por cluster. Para criar um sistema multilocatário usando o Neptune, use vários clusters ou particione logicamente os locatários em um único grafo e use a lógica do lado da aplicação para impor a separação. Por exemplo, adicione uma propriedade `tenantId` e inclua-a em cada consulta, da seguinte forma:

```
MATCH p=(n {tenantId:1})-[]->({tenantId:1}) RETURN p LIMIT 5)
```

O Neptune Serverless torna relativamente fácil implementar a multilocação usando vários clusters de banco de dados, sendo cada um escalado de forma independente e automática conforme necessário.

Suporte do Neptune para ferramentas do Neo4j

O Neptune oferece as seguintes alternativas às ferramentas do Neo4j:

- Neo4j Browser: o Neptune fornece blocos de anotações de grafos de código aberto que fornecem um IDE enfocado no desenvolvedor para executar consultas e visualizar os resultados.
- Neo4j Bloom: o Neptune é compatível com visualizações de grafos avançadas usando soluções de visualização de terceiros, como Graph-explorer, Tom Sawyer, Cambridge Intelligence, Graphistry, metaphacts e G.V().
- GraphQL: no momento, o Neptune é compatível com o GraphQL por meio de integrações AWS AppSync personalizadas. Consulte a postagem no blog Build a graph application with Amazon Neptune and AWS Amplify e o exemplo de projeto, Building Serverless Calorie tracker application with AWS AppSync and Amazon Neptune.
- NeoSemantics: o Neptune oferece suporte nativo ao modelo de dados do RDF, portanto, os clientes que desejam executar workloads do RDF são aconselhados a usar o suporte ao modelo do RDF do Neptune.

- Arrows.app: o Cypher criado ao exportar o modelo usando o comando export é compatível com o Neptune.
- Linkurious Ogma: um exemplo de integração com o Linkurious Ogma está disponível aqui.
- Spring Data Neo4j: no momento, não é compatível com o Neptune.
- Conector Spark do Neo4j: o conector Spark do Neo4j pode ser usado em um trabalho do Spark para se conectar ao Neptune usando o openCypher. Veja alguns exemplos de código e configuração da aplicação:

  Código de exemplo:

```
SparkSession spark = SparkSession
            .builder()
            .config("encryption.enabled", "true")
            .appName("Simple Application").config("spark.master",
 "local").getOrCreate();

Dataset<Row> df = spark.read().format("org.neo4j.spark.DataSource")
            .option("url", "bolt://(your cluster endpoint):8182")
            .option("encryption.enabled", "true")
            .option("query", "MATCH (n:airport) RETURN n")
            .load();

System.out.println("TOTAL RECORD COUNT: " + df.count());
spark.stop();
```

  Configuração da aplicação:

```
<dependency>
    <groupId>org.neo4j</groupId>
    <artifactId>neo4j-connector-apache-spark_2.12-4.1.0</artifactId>
    <version>4.0.1_for_spark_3</version>
</dependency>
```

Atributos e ferramentas do Neo4j não listados aqui

Se você estiver usando uma ferramenta ou um atributo não listado aqui, não temos certeza da compatibilidade com o Neptune ou com outros serviços da AWS contidos nele. Em caso de mais dúvidas, entre em contato com o AWS Support ou entre em contato com a equipe de sua conta.

Reformular consultas do Cypher para serem executadas no openCypher no Neptune

O openCypher é uma linguagem de consulta declarativa para grafos de propriedades originalmente desenvolvida pela Neo4j, que passou a ser de código aberto em 2015, e contribuiu para o projeto openCypher sob uma licença de código aberto Apache 2. Na AWS, acreditamos que o código aberto é bom para todos e estamos comprometidos em trazer o valor do código aberto aos nossos clientes e a excelência operacional da AWS para as comunidades de código aberto.

A sintaxe do OpenCypher está documentada na Cypher Query Language Reference, versão 9.

Como o openCypher contém um subconjunto da sintaxe e atributos da linguagem de consulta Cypher, alguns cenários de migração exigem a reformulação de consultas em formas compatíveis com o openCypher ou a análise de métodos alternativos para obter a funcionalidade desejada.

Esta seção contém recomendações para lidar com diferenças comuns, mas elas não são completas. É necessário testar as aplicações usando essas reformulações minuciosamente a fim de garantir que os resultados sejam os esperados.

Reformular funções de predicado **`None`**, **`All`** e **`Any`**

Essas funções não fazem parte da especificação do openCypher. É possível obter resultados comparáveis no openCypher usando a compreensão de lista.

Por exemplo, encontre todos os caminhos que vão do nó `Start` ao `End` , mas nenhuma jornada pode passar por um nó com a propriedade de classe `D`:

```
# Neo4J Cypher code
match p=(a:Start)-[:HOP*1..]->(z:End)
where none(node IN nodes(p) where node.class ='D')
return p

# Neptune openCypher code
match p=(a:Start)-[:HOP*1..]->(z:End)
where size([node IN nodes(p) where node.class = 'D']) = 0
return p
```

A compreensão de lista pode alcançar esses resultados da seguinte forma:

```
all  => size(list_comprehension(list)) = size(list)
```

```
any  => size(list_comprehension(list)) >= 1
none => size(list_comprehension(list)) = 0
```

Reformulando a função **`reduce()`** do Cypher em openCypher.

A função `reduce()` não faz parte da especificação do openCypher. Geralmente é usada para criar uma agregação de dados a partir de elementos em uma lista. Em muitos casos, é possível usar uma combinação de compreensão de lista e a cláusula `UNWIND` para obter resultados semelhantes.

Por exemplo, a consulta do Cypher a seguir encontra todos os aeroportos em caminhos com uma a três paradas entre Anchorage (ANC) e Austin (AUS) e exibe a distância total de cada caminho:

```
MATCH p=(a:airport {code: 'ANC'})-[r:route*1..3]->(z:airport {code: 'AUS'})
RETURN p, reduce(totalDist=0, r in relationships(p) | totalDist + r.dist) AS totalDist
ORDER BY totalDist LIMIT 5
```

Você pode reformular a mesma consulta em openCypher para Neptune da seguinte forma:

```
MATCH p=(a:airport {code: 'ANC'})-[r:route*1..3]->(z:airport {code: 'AUS'})
UNWIND [i in relationships(p) | i.dist] AS di
RETURN p, sum(di) AS totalDist
ORDER BY totalDist
LIMIT 5
```

Reformulando a cláusula FOREACH do Cypher em openCypher.

A cláusula FOREACH não faz parte da especificação do openCypher. Em geral, é usada para atualizar dados no meio de uma consulta, geralmente de agregações ou elementos em um caminho.

Como exemplo de caminho, encontre todos os aeroportos em um caminho com no máximo duas paradas entre Anchorage (ANC) e Austin (AUS) e defina uma propriedade visitada em cada um deles:

```
# Neo4J Example
MATCH p=(:airport {code: 'ANC'})-[*1..2]->({code: 'AUS'})
FOREACH (n IN nodes(p) | SET n.visited = true)

# Neptune openCypher
MATCH p=(:airport {code: 'ANC'})-[*1..2]->({code: 'AUS'})
WITH nodes(p) as airports
UNWIND airports as a
```

```
SET a.visited=true
```

Outro exemplo é:

```
# Neo4J Example
MATCH p=(start)-[*]->(finish)
WHERE start.name = 'A' AND finish.name = 'D'
FOREACH (n IN nodes(p) | SET n.marked = true)

# Neptune openCypher
MATCH p=(start)-[*]->(finish)
WHERE start.name = 'A' AND finish.name = 'D'
UNWIND nodes(p) AS n
SET n.marked = true
```

Reformulando procedimentos APOC do Neo4j no Neptune.

Os exemplos abaixo usam o openCypher para substituir alguns dos procedimentos APOC mais usados. Esses exemplos são apenas para referência e têm como objetivo fornecer algumas sugestões sobre como lidar com cenários comuns. Na prática, cada aplicação é diferente e você precisará criar suas próprias estratégias para fornecer todas as funcionalidades de que precisa.

Procedimentos de reformulação de **`apoc.export`**

O Neptune fornece uma variedade de opções para exportações completas com base em grafos e consultas em vários formatos de saída, como CSV e JSON, usando o utilitário neptune-export (consulte Exportar dados de um cluster de banco de dados do Neptune).

Procedimentos de reformulação de **`apoc.schema`**

O Neptune não tem esquemas, índices nem restrições explicitamente definidos, portanto, muitos procedimentos `apoc.schema` não são mais necessários. Veja estes exemplos:

- `apoc.schema.assert`
- `apoc.schema.node.constraintExists`
- `apoc.schema.node.indexExists`,
- `apoc.schema.relationship.constraintExists`
- `apoc.schema.relationship.indexExists`
- `apoc.schema.nodes`

- `apoc.schema.relationships`

O openCypher no Neptune é compatível com a recuperação de valores semelhantes aos dos procedimentos, conforme mostrado abaixo, mas pode ter problemas de desempenho em grafos maiores, pois isso requer a verificação de uma grande parte do grafo para exibir a resposta.

```
# openCypher replacement for apoc.schema.properties.distinct
MATCH (n:airport)
RETURN DISTINCT n.runways
```

```
# openCypher replacement for apoc.schema.properties.distinctCount
MATCH (n:airport)
RETURN DISTINCT n.runways, count(n.runways)
```

Alternativas aos procedimentos **`apoc.do`**

Esses procedimentos são usados para fornecer uma execução condicional de consultas que seja fácil de implementar usando outras cláusulas do openCypher. No Neptune, existem pelo menos duas maneiras de obter um comportamento semelhante:

- Uma delas é combinar os recursos de compreensão de listas do openCypher com a cláusula `UNWIND`.
- Outra forma é usar as etapas choose() e coalesce() no Gremlin.

Exemplos dessas abordagens são mostrados abaixo.

Alternativas para apoc.do.when

```
# Neo4J Example
MATCH (n:airport {region: 'US-AK'})
CALL apoc.do.when(
 n.runways>=3,
 'SET n.is_large_airport=true RETURN n',
 'SET n.is_large_airport=false RETURN n',
 {n:n}
) YIELD value
WITH collect(value.n) as airports
RETURN size([a in airports where a.is_large_airport]) as large_airport_count,
size([a in airports where NOT a.is_large_airport]) as small_airport_count
```

```
# Neptune openCypher
MATCH (n:airport {region: 'US-AK'})
WITH n.region as region, collect(n) as airports
WITH [a IN airports where a.runways >= 3] as large_airports,
[a IN airports where a.runways < 3] as small_airports, airports
UNWIND large_airports as la
SET la.is_large_airport=true
WITH DISTINCT small_airports, airports
UNWIND small_airports as la
    SET la.small_airports=true
WITH DISTINCT airports
RETURN size([a in airports where a.is_large_airport]) as large_airport_count,
size([a in airports where NOT a.is_large_airport]) as small_airport_count

#Neptune Gremlin using choose()
g.V().
  has('airport', 'region', 'US-AK').
  choose(
    values('runways').is(lt(3)),
    property(single, 'is_large_airport', false),
    property(single, 'is_large_airport', true)).
  fold().
  project('large_airport_count', 'small_airport_count').
    by(unfold().has('is_large_airport', true).count()).
    by(unfold().has('is_large_airport', false).count())

 #Neptune Gremlin using coalesce()
g.V().
  has('airport', 'region', 'US-AK').
  coalesce(
    where(values('runways').is(lt(3))).
    property(single, 'is_large_airport', false),
    property(single, 'is_large_airport', true)).
  fold().
  project('large_airport_count', 'small_airport_count').
    by(unfold().has('is_large_airport', true).count()).
    by(unfold().has('is_large_airport', false).count())
```

Alternativas para apoc.do.case

```
# Neo4J Example
```

```
MATCH (n:airport {region: 'US-AK'})
CALL apoc.case([
 n.runways=1, 'RETURN "Has one runway" as b',
 n.runways=2, 'RETURN "Has two runways" as b'
 ],
 'RETURN "Has more than 2 runways" as b'
) YIELD value
RETURN {type: value.b,airport: n}

# Neptune openCypher
MATCH (n:airport {region: 'US-AK'})
WITH n.region as region, collect(n) as airports
WITH [a IN airports where a.runways =1] as single_runway,
[a IN airports where a.runways =2] as double_runway,
[a IN airports where a.runways >2] as many_runway
UNWIND single_runway as sr
    WITH {type: "Has one runway",airport: sr} as res, double_runway, many_runway
WITH DISTINCT double_runway as double_runway, collect(res) as res, many_runway
UNWIND double_runway as dr
    WITH {type: "Has two runways",airport: dr} as two_runways, res, many_runway
WITH collect(two_runways)+res as res, many_runway
UNWIND many_runway as mr
    WITH {type: "Has more than 2 runways",airport: mr} as res2, res, many_runway
WITH collect(res2)+res as res
UNWIND res as r
RETURN r

#Neptune Gremlin using choose()
g.V().
  has('airport', 'region', 'US-AK').
  project('type', 'airport').
    by(
      choose(values('runways')).
        option(1, constant("Has one runway")).
        option(2, constant("Has two runways")).
        option(none, constant("Has more than 2 runways"))).
    by(elementMap())

 #Neptune Gremlin using coalesce()
 g.V().
  has('airport', 'region', 'US-AK').
  project('type', 'airport').
    by(
      coalesce(
```

```
        has('runways', 1).constant("Has one runway"),
        has('runways', 2).constant("Has two runways"),
        constant("Has more than 2 runways"))).
    by(elementMap())
```

Alternativas às propriedades baseadas em listas

No momento, o Neptune não aceita o armazenamento de propriedades baseadas em listas. No entanto, resultados semelhantes podem ser obtidos armazenando os valores da lista como uma string separada por vírgula e, depois, usando as funções `join()` e `split()` para construir e desconstruir a propriedade da lista.

Por exemplo, se quisermos salvar uma lista de tags como uma propriedade, poderemos usar o exemplo de reformulação, que mostra como recuperar uma propriedade separada por vírgula e, depois, usar as funções `split()` e `join()` com a compreensão de lista para obter resultados comparáveis:

```
# Neo4j Example (In this example, tags is a durable list of string.
MATCH (person:person {name: "TeeMan"})
WITH person, [tag in person.tags WHERE NOT (tag IN ['test1', 'test2', 'test3'])] AS
 newTags
SET person.tags = newTags
RETURN person

# Neptune openCypher
MATCH (person:person {name: "TeeMan"})
WITH person, [tag in split(person.tags, ',') WHERE NOT (tag IN ['test1', 'test2',
 'test3'])] AS newTags
SET person.tags = join(newTags,',')
RETURN person
```

Recursos para migrar do Neo4j para o Neptune

O Neptune oferece várias ferramentas e recursos que podem auxiliar no processo de migração.

Ferramentas para ajudar a migrar do Neo4j para o Neptune

- A CheatSheet do openCypher.
- neo4j-to-neptune: um utilitário de linha de comando para migrar dados do Neo4j para o Neptune.
- fully-automated-neo4j-to-neptune: uma aplicação de CDK que mostra como migrar bancos de dados Neo4j simples para o Amazon Neptune.
- csv para neptune-bulk-format: essa ferramenta adota uma abordagem baseada em configuração para reformatar um ou mais arquivos CSV em um formato compatível de carregamento em massa do Neptune.

Publicações no blog

- Change data capture from Neo4j to Amazon Neptune using Amazon Managed Streaming for Apache Kafka de Sanjeet Sahay (22 de junho de 2020).
- Migrating a Neo4j graph database to Amazon Neptune with a fully automated utility de Sanjeet Sahay (13 de abril de 2020).

Migrar um grafo existente de um servidor Apache TinkerPop Gremlin para o Amazon Neptune

Se você tiver dados de grafos em um servidor Apache TinkerPop Gremlin que gostaria de migrar para o Amazon Neptune, siga estas etapas:

1. Exporte os dados do servidor Gremlin para o Amazon Simple Storage Service (Amazon S3).
2. Converta os dados exportados em um formato CSV que o carregador em massa do Neptune possa importar.
3. Usando Carregador em massa do Neptune, importe os dados para um cluster de banco de dados do Neptune que você preparou.
4. Modifique a aplicação existente para se conectar ao endpoint do Gremlin do Neptune e faça as alterações necessárias para se adequar às diferenças de implementação do Gremlin do Neptune.

Migrar um grafo existente de um armazenamento triplo do RDF para o Amazon Neptune

Se você tiver dados de grafos em um RDF/SPARQL para migrar para o Amazon Neptune, siga estas etapas:

1. Exporte os dados do armazenamento triplo do RDF.
2. Converta os dados exportados em um formato que o carregador em massa do Neptune possa importar.
3. Armazene os dados a serem importados no Amazon Simple Storage Service (Amazon S3).
4. Usando Carregador em massa do Neptune, importe os dados do Amazon S3 para um cluster de banco de dados do Neptune que você preparou.
5. Modifique a aplicação existente para se conectar ao endpoint SPARQL do Neptune.

Se quiser experimentar a migração de dados CSV do grafo de propriedades para o RDF, você pode usar o conversor de CSV para RDF do Amazon Neptune.

Usar AWS Database Migration Service (AWS DMS) para migrar de um banco de dados relacional ou NoSQL para o Amazon Neptune

O AWS Database Migration Service (AWS DMS) é um serviço em nuvem que facilita a migração de bancos de dados relacionais, data warehouses, bancos de dados NoSQL e outros tipos de armazenamentos de dados. Se você tiver dados de grafos armazenados em um dos bancos AWS DMS de dados relacionais ou NoSQL compatíveis com , o AWS DMS poderá ajudar você a migrar para o Neptune de forma rápida e segura, sem exigir tempo de inatividade do banco de dados atual. Para mais detalhes, consulte Usando AWS Database Migration Service para carregar dados no Amazon Neptune a partir de um armazenamento de dados diferente.

O fluxo de dados de migração usado o AWS DMS é o seguinte:

- Crie um objeto de mapeamento de tabela do AWS DMS. Esse objeto JSON especifica quais tabelas devem ser lidas do banco de dados de origem e em que ordem e como as colunas são chamadas. Ele também pode filtrar as linhas que estão sendo copiadas e fornecer transformações de valor simples, como converter para letras minúsculas ou arredondar.
- Crie uma `GraphMappingConfig` do Neptune para especificar como os dados extraídos do banco de dados de origem devem ser carregados no Neptune.
  - Para dados do RDF (consultados usando o SPARQL), o `GraphMappingConfig` é escrito na linguagem de mapeamento R2RML padrão do W3.
  - Para dados de grafos de propriedades (consultados usando o Gremlin), a `GraphMappingConfig` é um objeto JSON, descrito em GraphMappingConfig Layout para dados de gráfico de propriedade/Gremlin.
- Crie uma instância de replicação do AWS DMS na mesma VPC do cluster de banco de dados do Neptune para realizar a migração.
- Crie um bucket do Amazon S3 a ser usado como armazenamento intermediário para preparar os dados que estão sendo migrados.
- Execute a tarefa de migração do AWS DMS.

Consulte Usando AWS Database Migration Service para carregar dados no Amazon Neptune a partir de um armazenamento de dados diferente para obter os detalhes e também a postagem em quatro partes no blog de Chris Smith, “Populating your graph in Amazon Neptune from a relational database using AWS Database Migration Service (DMS):”

- Part 1: Setting the stage

- Part 2: Designing the property graph model
- Part 3: Designing the RDF Model
- Part 4: Putting it all together

Migrar do Blazegraph para o Amazon Neptune

Se você tiver um grafo no armazenamento triplo do RDF Blazegraph de código aberto, poderá migrar os dados de grafos para o Amazon Neptune usando as seguintes etapas:

- Provisione a infraestrutura da AWS. Comece provisionando a infraestrutura Neptune necessária usando um modelo do AWS CloudFormation (consulte Criar um cluster de banco de dados).
- Exporte dados do Blazegraph. Existem dois métodos principais para exportar dados do Blazegraph, ou seja, usar consultas SPARQL CONSTRUCT ou o utilitário Export do Blazegraph.
- Importe os dados para o Neptune. Depois, é possível carregar os arquivos de dados exportados no Neptune usando a bancada de trabalho do Neptune e Carregador em massa do Neptune.

Em geral, essa abordagem também é aplicável à migração de outros bancos de dados triplestore do RDF.

Compatibilidade do Blazegraph com o Neptune

Antes de migrar os dados de grafos para o Neptune, há várias diferenças significativas entre o Blazegraph e o Neptune que você deve conhecer. Essas diferenças podem exigir alterações nas consultas, na arquitetura da aplicação ou em ambas, ou até mesmo tornar a migração impraticável:

- **`Full-text search`**: no Blazegraph, é possível usar os recursos de pesquisa interna ou externa de texto completo por meio de uma integração com o Apache Solr. Se você usar qualquer um desses atributos, informe-se sobre as atualizações mais recentes dos atributos de pesquisa de texto completo compatíveis com o Neptune. Consulte Pesquisa de texto completo do Neptune.
- **`Query hints`**: tanto o Blazegraph quanto o Neptune estendem o SPARQL usando o conceito de dicas de consulta. Durante uma migração, é necessário migrar todas as dicas de consulta usadas. Para obter informações sobre as dicas de consulta mais recentes que o Neptune aceita, consulte Dicas de consulta do SPARQL.
- Inferência: o Blazegraph é compatível com inferência como uma opção configurável no modo de triplos, mas não no modo de quádruplos. O Neptune ainda não é compatível com inferência.
- Pesquisa geoespacial: o Blazegraph é compatível com a configuração de namespaces que permitem suporte geoespacial. Esse atributo ainda não está disponível no Neptune.
- Multilocação: o Blazegraph é compatível com multilocação em um único banco de dados. No Neptune, a multilocação é compatível armazenando dados em grafos nomeados e usando as

cláusulas USING NAMED para consultas do SPARQL ou criando um cluster de banco de dados separado para cada locatário.

- Federação: no momento, o Neptune é compatível com a federação SPARQL 1.1 em locais acessíveis à instância do Neptune, como na VPC privada, entre VPCs ou em endpoints externos da Internet. Dependendo da configuração específica e dos endpoints de federação necessários, talvez seja necessária alguma configuração de rede adicional.
- Extensões de padrões do Blazegraph: o Blazegraph inclui várias extensões para os padrões do SPARQL e da API REST, enquanto o Neptune é compatível apenas com as próprias especificações dos padrões. Isso pode exigir alterações na aplicação ou dificultar a migração.

Provisionar a infraestrutura da AWS para o Neptune

Embora seja possível construir a infraestrutura da AWS necessária manualmente por meio do AWS Management Console ou da AWS CLI, geralmente é mais conveniente usar um modelo do CloudFormation, conforme descrito abaixo:

Provisionando o Neptune com um modelo do CloudFormation:

1. Acesse Usando uma AWS CloudFormation pilha para criar um cluster de banco de dados Neptune.
2. Selecione Iniciar pilha na região de preferência.
3. Defina os parâmetros necessários (nome da pilha e `EC2SSHKeyPairName`). Defina também os seguintes parâmetros opcionais para facilitar o processo de migração:
   - Defina `AttachBulkloadIAMRoleToNeptuneCluster` como verdadeiro. Esse parâmetro permite criar e anexar o perfil apropriado do IAM ao cluster para permitir o carregamento em massa de dados.
   - Defina `NotebookInstanceType` como o tipo de instância de sua preferência. Esse parâmetro cria uma pasta de trabalho do Neptune que você usa para executar o carregamento em massa no Neptune e validar a migração.
4. Escolha Next (Próximo).
5. Defina as outras opções de pilha desejadas.
6. Escolha Next (Próximo).
7. Analise as opções e marque as duas caixas de seleção para confirmar que o AWS CloudFormation pode exigir recursos adicionais.

8. Selecione Criar pilha.

O processo de criação da pilha pode levar alguns minutos.

Exportar dados do Blazegraph

A próxima etapa é exportar dados do Blazegraph em um formato compatível com o carregador em massa do Neptune.

Dependendo de como os dados são armazenados no Blazegraph (triplos ou quádruplos) e de quantos grafos nomeados estão em uso, o Blazegraph pode exigir que você execute o processo de exportação várias vezes e gere vários arquivos de dados:

- Se os dados forem armazenados em triplos, você precisará executar uma exportação para cada grafo nomeado.
- Se os dados forem armazenados como quads, você poderá optar por exportar dados no formato N-Quads ou exportar cada grafo nomeado em um formato de triplos.

Abaixo, supomos que você exporte um único namespace como N-Quads, mas você pode repetir o processo para namespaces adicionais ou formatos de exportação desejados.

Se você precisar que o Blazegraph esteja on-line e disponível durante a migração, use as consultas SPARQL CONSTRUCT. Isso requer que você instale, configure e execute uma instância do Blazegraph com um endpoint do SPARQL acessível.

Se for necessário que o Blazegraph esteja on-line, use o utilitário BlazeGraph Export. Para fazer isso, é necessário baixar o Blazegraph, e o arquivo de dados e de configuração precisam estar acessíveis, mas o servidor não precisa estar em execução.

Exportar dados do Blazegraph usando SPARQL CONSTRUCT

SPARQL CONSTRUCT é um atributo do SPARQL que gera um grafo do RDF correspondente ao modelo de consulta especificado. Para esse caso de uso, você o usa para exportar os dados um namespace por vez, usando uma consulta como a seguinte:

```
CONSTRUCT WHERE { hint:Query hint:analytic "true" . hint:Query
 hint:constructDistinctSPO "false" . ?s ?p ?o }
```

Embora existam outras ferramentas do RDF para exportar esses dados, a maneira mais fácil de executar essa consulta é usando o endpoint da API REST fornecido pelo Blazegraph. O seguinte script demonstra como usar um script Python (3.6+) para exportar dados como N-Quads:

```
import requests

# Configure the URL here: e.g. http://localhost:9999/sparql
url = "http://localhost:9999/sparql"
payload = {'query': 'CONSTRUCT WHERE { hint:Query hint:analytic "true" . hint:Query
 hint:constructDistinctSPO "false" . ?s ?p ?o }'}
# Set the export format to be n-quads
headers = {
'Accept': 'text/x-nquads'
}
# Run the http request
response = requests.request("POST", url, headers=headers, data = payload, files = [])
#open the file in write mode, write the results, and close the file handler
f = open("export.nq", "w")
f.write(response.text)
f.close()
```

Se os dados forem armazenados como triplos, você precisará alterar o parâmetro de cabeçalho `Accept` para exportar dados em um formato apropriado (N-Triples, RDF/XML ou Turtle) usando os valores especificados no repositório do Blazegraph no GitHub.

Usar o utilitário de exportação Blazegraph para exportar dados

O Blazegraph contém um método utilitário para exportar dados, ou seja, a classe `ExportKB`. O `ExportKB` facilita a exportação de dados do Blazegraph, mas, ao contrário do método anterior, exige que o servidor esteja off-line enquanto a exportação está sendo executada. Isso o torna o método ideal para ser usado quando você pode colocar o Blazegraph off-line durante a migração, ou a migração pode ocorrer a partir de um backup dos dados.

Você executa o utilitário a partir de uma linha de comando Java em uma máquina que tem o Blazegraph instalado, mas não está em execução. A maneira mais fácil de executar esse comando é baixar a versão mais recente do blazegraph.jar localizada no GitHub. A execução desse comando requer vários parâmetros:

- **`log4j.primary.configuration`**: a localização do arquivo de propriedades log4j.
- **`log4j.configuration`**: a localização do arquivo de propriedades log4j.

- **`output`**: o diretório de saída dos dados exportados. Os arquivos estão localizados como `tar.gz` em um subdiretório chamado conforme documentado na base de conhecimento.
- **`format`**: o formato de saída desejado seguido pela localização do arquivo `RWStore.properties`. Se você estiver trabalhando com triplos, precisará alterar o parâmetro `-format` para `N-Triples`, `Turtle` ou `RDF/XML`.

Por exemplo, se você tiver o arquivo de diário e os arquivos de propriedades do Blazegraph, exporte os dados como N-Quads usando o seguinte código:

```
java -cp blazegraph.jar \
    com.bigdata.rdf.sail.ExportKB \
    -outdir ~/temp/ \
    -format N-Quads \
    ./RWStore.properties
```

Se a exportação for bem-sucedida, a saída será semelhante à seguinte:

```
Exporting kb as N-Quads on /home/ec2-user/temp/kb
Effective output directory: /home/ec2-user/temp/kb
Writing /home/ec2-user/temp/kb/kb.properties
Writing /home/ec2-user/temp/kb/data.nq.gz
Done
```

Criar um bucket do Amazon Simple Storage Service (Amazon S3) e copiar os dados exportados nele

Depois de exportar os dados do Blazegraph, crie um bucket do Amazon Simple Storage Service (Amazon S3) na mesma região do cluster de banco de dados do Neptune de destino para o carregador em massa do Neptune usar para importar os dados.

Para obter instruções sobre como criar um bucket do Amazon S3, consulte How do I create an S3 Bucket? no Guia do usuário do Amazon Simple Storage Service e Examples of creating a bucket no Guia do usuário do Amazon Simple Storage Service.

Para obter instruções sobre como copiar os arquivos de dados exportados para o novo bucket do Amazon S3, consulte Uploading an object to a bucket no Guia do usuário do Amazon Simple Storage Service ou Using high-level (s3) commands with the AWS CLI. Você também pode usar o código Python como o seguinte para copiar os arquivos um por um:

```
import boto3

region = 'region name'
bucket_name = 'bucket name'
s3 = boto3.resource('s3')
s3.meta.client.upload_file('export.nq', bucket_name, 'export.nq')
```

Usar o carregador em massa do Neptune para importar os dados para o Neptune

Depois de exportar os dados do Blazegraph e copiá-los em um bucket do Amazon S3, estará tudo pronto para a importação dos dados para o Neptune. O Neptune tem um carregador em massa que carrega dados com maior rapidez e menor sobrecarga do que realizar operações de carregamento usando SPARQL. O processo do carregador em massa é iniciado por uma chamada à API do endpoint do carregador para carregar os dados armazenados no bucket do S3 identificado no Neptune.

Embora seja possível fazer isso com uma chamada direta para o endpoint REST do carregador, você deve ter acesso à VPC privada na qual a instância de destino do Neptune é executada. Você pode configurar um bastion host, SSH nessa máquina e executar o comando cURL, mas usar a bancada de trabalho do Neptune é mais fácil.

A bancada de trabalho do Neptune é um caderno Jupyter pré-configurado executado como um bloco de anotações Amazon SageMaker, com várias magias de bloco de anotações específicas do Neptune instaladas. Essas magias simplificam as operações comuns do Neptune, como conferir o status do cluster, executar percursos do SPARQL e do Gremlin e executar uma operação de carregamento em massa.

Para iniciar o processo de carregamento em massa, use a magia `%load`, que fornece uma interface para executar o Comando do carregador do Neptune:

1. Faça login no Console de Gerenciamento da AWS e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. Selecione aws-neptune-blazegraph-to-neptune.
3. Escolha Abrir bloco de anotações.
4. Na instância em execução do Jupyter, selecione um caderno existente ou crie um usando o kernel do Python 3.
5. No bloco de anotações, abra uma célula, insira `%load` e execute a célula.

6. Defina os parâmetros para o carregador em massa:

   a. Em Fonte, insira a localização de um arquivo de origem a ser importado: `s3://{bucket_name}/{file_name}`.

   b. Em Formato, escolha o formato apropriado, que neste exemplo é `nquads`.

   c. Em Carregar ARN, insira o ARN do perfil `IAMBulkLoad` (essas informações estão localizadas no console do IAM em Perfis).

7. Selecione Submit (Enviar).

O resultado contém o status da solicitação. As cargas em massa geralmente são processos de longa duração, então a resposta não significa que a carga foi concluída, apenas que ela começou. Essas informações de status são atualizadas periodicamente até relatar que o trabalho foi concluído.

> **Note**
>
> Essas informações também estão disponíveis na postagem no blog, Moving to the cloud: Migrating Blazegraph to Amazon Neptune.

Carregar dados no Amazon Neptune

Existem várias maneiras diferentes de carregar dados de grafo no Amazon Neptune:

- Se for necessário carregar somente uma quantidade relativamente pequena de dados, será possível usar consultas como instruções `INSERT` do SPARQL ou `addV` do Gremlin e etapas `addE`.
- É possível aproveitar Carregador em massa do Neptune para ingerir grandes quantidades de dados que residem em arquivos externos. O comando do carregador em massa é mais rápido e tem menos sobrecarga do que os comandos query-language. Ele é otimizado para grandes conjuntos de dados e oferece suporte a dados do RDF (Resource Description Framework) e do Gremlin.
- Você pode usar AWS Database Migration Service (AWS DMS) para importar dados de outros armazenamentos de dados (consulte Usando AWS Database Migration Service para carregar dados no Amazon Neptune a partir de um armazenamento de dados diferente e o Guia AWS Database Migration Service do Usuário).
- Por fim, é possível usar a etapa `g.io(URL).read()` do Gremlin para ler arquivos de dados em GraphML (um formato XML), GraphSon (um formato JSON) e outros formatos. Consulte a TinkerPopdocumentação para obter detalhes.

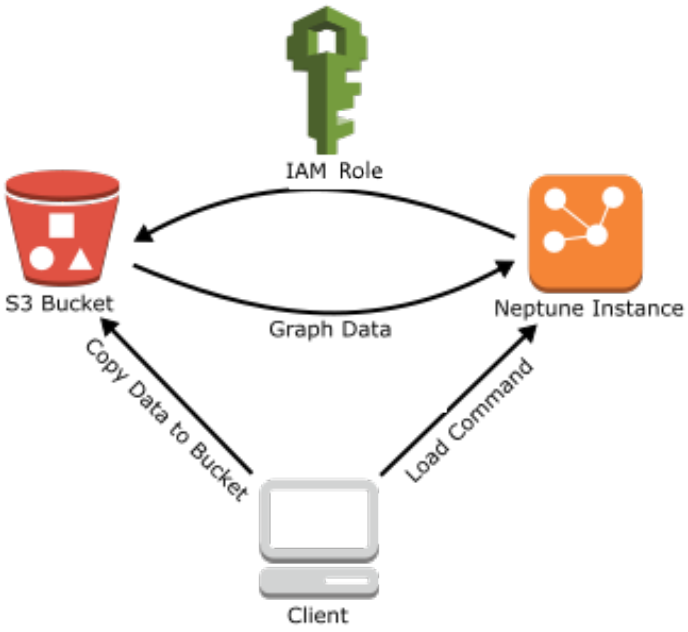
Tópicos

Usar o carregador em massa do Amazon Neptune para ingerir dados

O Amazon Neptune fornece um comando `Loader` para carregar dados de arquivos externos diretamente em um cluster de banco de dados do Neptune. É possível usar esse comando em vez de executar um grande número de declarações `INSERT`, etapas `addV` e `addE` ou outras chamadas de API.

O comando Loader do Neptune é mais rápido, tem menor sobrecarga, é otimizado para grandes conjuntos de dados e é compatível com dados do Gremlin e do RDF (Resource Description Framework) usados pelo SPARQL.

O diagrama a seguir mostra uma visão geral do processo de carga:



Veja as etapas do processo de carregamento:

1. Copie os arquivos de dados em um bucket do Amazon Simple Storage Service (Amazon S3).
2. Crie uma função do IAM com acesso de leitura e lista no bucket.
3. Crie um endpoint da VPC do Amazon S3.
4. Inicie o carregador do Neptune enviando uma solicitação à instância de banco de dados do Neptune por meio de HTTP.
5. A instância de banco de dados do Neptune assume o perfil do IAM para carregar os dados do bucket.

> **Note**
>
> Será possível carregar dados criptografados do Amazon S3 se eles tiverem sido criptografados usando `SSE-S3` do Amazon S3 ou o modo `SSE-KMS`, desde que o perfil utilizado para carregamento em massa tenha acesso ao objeto do Amazon S3 e, no caso da

SSE-KMS, `kms:decrypt`. O Neptune pode personificar suas credenciais e emitir chamadas do `s3:getObject` em seu nome.
No entanto, o Neptune no momento não é compatível com o carregamento de dados criptografados usando o modo `SSE-C`.

As seções a seguir fornecem instruções para preparar e carregar dados no Neptune.

Tópicos

Pré-requisitos: perfil do IAM e acesso ao Amazon S3

Carregar dados de um bucket do Amazon Simple Storage Service (Amazon S3) requer AWS Identity and Access Management uma função (IAM) que tenha acesso ao bucket. O Amazon Neptune assume essa função de carregar os dados.

> **Note**
>
> Você poderá carregar dados criptografados do Amazon S3 se ele foi criptografado usando o modo `SSE-S3` do Amazon S3. Nesse caso, o Neptune pode personificar suas credenciais e emitir chamadas `s3:getObject` em seu nome.
> Você também pode carregar dados criptografados do Amazon S3 que foram criptografados usando o modo `SSE-KMS`, desde que o perfil do IAM inclua as permissões necessárias para acessar o AWS KMS. Sem AWS KMS as permissões adequadas, a operação de carregamento em massa falha e retorna uma `LOAD_FAILED` resposta.
> No momento, o Neptune não é compatível com o carregamento de dados criptografados do Amazon S3 usando o modo `SSE-C`.

As seções a seguir mostram como usar uma política do IAM gerenciada para criar um perfil do IAM para acessar recursos do Amazon S3 e, depois, associar o perfil ao seu cluster do Neptune.

Tópicos

> **Note**
>
> Essas instruções exigem que você tenha acesso ao console do IAM e permissões para gerenciar perfis e políticas do IAM. Para obter mais informações, consulte Permissões para trabalhar no console AWS de gerenciamento no Guia do usuário do IAM.
> O console do Amazon Neptune exige que o usuário tenha as seguintes permissões do IAM para associar o perfil ao cluster do Neptune:
>
> ```
> iam:GetAccountSummary on resource: *
> iam:ListAccountAliases on resource: *
> iam:PassRole on resource: * with iam:PassedToService restricted to
>  rds.amazonaws.com
> ```

Criar um perfil do IAM para permitir que o Amazon Neptune acesse os recursos do Amazon S3

Use a política gerenciada do IAM `AmazonS3ReadOnlyAccess` para criar um perfil do IAM que permitirá ao Amazon Neptune acessar os recursos do Amazon S3.

Como criar um perfil do IAM que permita ao Neptune acessar o Amazon S3

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, selecione Roles.
3. Escolha Criar Perfil.
4. Em Serviço da AWS , escolha S3.
5. Selecione Next: Permissions (Próximo: permissões).
6. Use a caixa de filtro para filtrar pelo termo S3 e marque a caixa ao lado do ReadOnlyAmazonS3 Access.

> **Note**
>
> Essa política concede as permissões `s3:Get*` e `s3:List*` a todos os buckets. As próximas etapas restringem o acesso à função usando a política de confiança.
> O carregador requer apenas as permissões `s3:Get*` e `s3:List*` para o bucket a partir do qual você está realizando o carregamento, para que também possa restringir essas permissões pelo recurso do Amazon S3.
> Se o bucket do S3 foi criptografado, você precisa adicionar permissões `kms:Decrypt`

7. Selecione Next: Review (Próximo: revisar).
8. Defina Nome do perfil como um nome para o perfil do IAM, por exemplo, `NeptuneLoadFromS3`. Você também pode adicionar um valor Descrição do perfil, como: "Permitir que o Neptune acesse os recursos do Amazon S3 em seu nome".
9. Selecione Criar função.
10. No painel de navegação, escolha Perfis.
11. No campo Search (Pesquisar), digite o nome da função que você criou e selecione a função quando ela aparecer na lista.
12. Na guia Trust Relationships (Relacionamentos confiáveis), selecione Edit Trust Relationship (Editar relacionamento confiável).
13. No campo de texto, cole a política de confiança a seguir.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "rds.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

14. Selecione Atualizar política de confiança.

15. Siga as etapas em Adicionar o perfil do IAM a um cluster do Amazon Neptune.

Adicionar o perfil do IAM a um cluster do Amazon Neptune

Use o console para adicionar o perfil do IAM a um cluster do Amazon Neptune. Isso permite que qualquer instância de banco de dados do Neptune no cluster assuma o perfil e carregue do Amazon S3.

> **Note**
>
> O console do Amazon Neptune exige que o usuário tenha as seguintes permissões do IAM para associar o perfil ao cluster do Neptune:
>
> ```
> iam:GetAccountSummary on resource: *
> iam:ListAccountAliases on resource: *
> iam:PassRole on resource: * with iam:PassedToService restricted to
>  rds.amazonaws.com
> ```

Como adicionar um perfil do IAM a um cluster do Amazon Neptune

1. Faça login no AWS Management Console e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. No painel de navegação, escolha Bancos de dados.
3. Escolha o identificador do cluster que você deseja modificar.
4. Escolha a guia Conectividade e segurança.
5. Na seção Funções do IAM, escolha a função que você criou na seção anterior.
6. Escolha Add role (adicionar função).
7. Aguarde até que o perfil do IAM se torne acessível para o cluster para usá-lo.

Criar o endpoint da VPC do Amazon S3

O carregador do Neptune exige um endpoint da VPC do tipo Gateway para Amazon S3.

Para configurar o acesso ao Amazon S3

1. Faça login AWS Management Console e abra o console da Amazon VPC em https://console.aws.amazon.com/vpc/.
2. No painel de navegação, escolha Endpoints.
3. Escolha Create Endpoint (Criar endpoint).
4. Selecione o Nome do serviço `com.amazonaws.region.s3` para o endpoint do tipo Gateway.

   > **Note**
   >
   > Se a região aqui estiver incorreta, verifique se a região do console está correta.

5. Selecione a VPC que contém a instância de banco de dados do Neptune (ela está listada para a instância de banco de dados no console do Neptune).
6. Marque a caixa de seleção ao lado das tabelas de rotas associadas às sub-redes relacionados ao cluster. Se tiver apenas uma tabela de rotas, você deverá selecionar essa caixa.
7. Escolha Criar Endpoint.

Para obter informações sobre criação do endpoint, consulte VPC Endpoints no Guia do usuário da Amazon VPC. Para obter informações sobre as limitações dos endpoints da VPC, consulte VPC Endpoints for Amazon S3.

Próximos Passos

Agora que concedeu acesso ao bucket do Amazon S3, você pode se preparar para carregar os dados. Para obter informações sobre os formatos compatíveis, consulte Formatos de dados de carga.

Encadear perfis do IAM no Amazon Neptune

> **Important**
>
> O novo atributo de carregamento em massa entre contas introduzido na versão 1.2.1.0.R3 do mecanismo, que utiliza o encadeamento de perfis do IAM, pode, em alguns casos, causar degradação do desempenho de carregamento em massa. Como resultado, as atualizações das versões do mecanismo compatíveis com esse atributo foram temporariamente suspensas até que o problema seja resolvido.

Ao associar um perfil ao cluster, seu cluster poderá assumir esse perfil para obter acesso aos dados armazenados no Amazon S3. A partir da versão 1.2.1.0.R3 do mecanismo, se esse perfil não tiver acesso a todos os recursos necessários, você poderá encadear um ou mais perfis adicionais que o cluster poderá assumir para obter acesso a outros recursos. Cada perfil na cadeia assume o próximo perfil na cadeia, até que o cluster assuma o perfil no final da cadeia.

Para encadear perfis, você estabelece uma relação de confiança entre eles. Por exemplo, para encadear o `RoleB` com o `RoleA`, o `RoleA` precisa ter uma política de permissões que o possibilite assumir o `RoleB` e `RoleB` deve ter uma política de confiança que o permita transmitir as permissões de volta ao `RoleA`. Para obter mais informações, consulte Using IAM roles.

O primeiro perfil da cadeia deve ser associado ao cluster que está carregando dados.

O primeiro perfil e cada perfil subsequente que assumir o perfil seguinte na cadeia devem ter:

- Uma política que inclua uma declaração específica com o efeito `Allow` na ação `sts:AssumeRole`.
- O nome do recurso da Amazon (ARN) do próximo perfil em um elemento `Resource`.

> **Note**
>
> O bucket de destino do Amazon S3 deve estar na mesma AWS região do cluster.

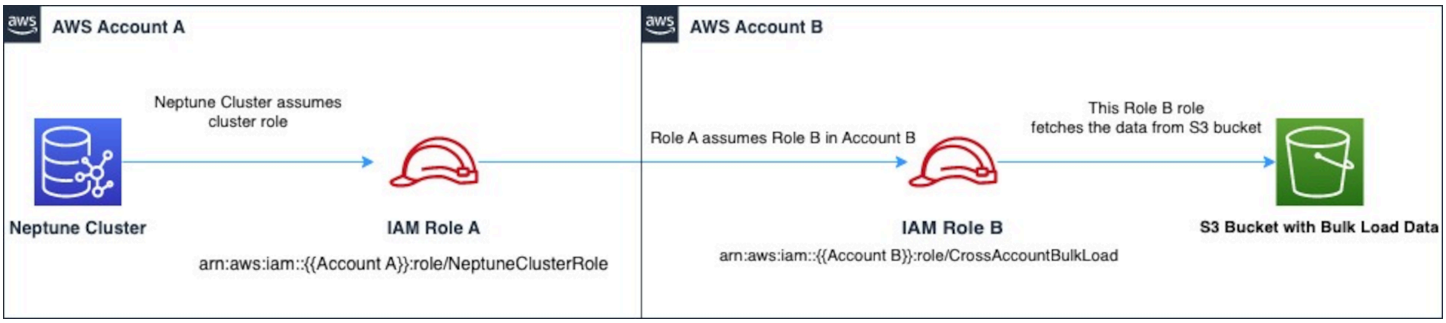
Acesso entre contas usando perfis encadeados

É possível conceder acesso entre contas encadeando um perfil ou perfis pertencentes a outra conta. Quando o cluster assume temporariamente um perfil pertencente a outra conta, ele pode obter acesso aos recursos dessa conta.

Por exemplo, suponha que a Conta A queira acessar dados em um bucket do Amazon S3 que pertença à Conta B:

- A conta A cria uma função AWS de serviço para Neptune `RoleA` chamada e a anexa a um cluster.
- A Conta B cria um perfil chamado `RoleB` que é autorizado a acessar os dados no bucket da Conta B.
- A Conta A associa uma política de permissões ao `RoleA` que permite a ele assumir o `RoleB`.

- A Conta B atribui uma política de confiança ao `RoleB` que permite a ele transmitir as permissões de volta ao `RoleA`.
- Para acessar os dados no bucket da Conta B, a Conta A executa um comando do carregador usando um parâmetro `iamRoleArn` que encadeia `RoleA` e `RoleB`. Durante a operação do carregador, `RoleA` assume temporariamente `RoleB` para acessar o bucket do Amazon S3 na Conta B.



Por exemplo, o `RoleA` teria uma política de confiança que estabelecesse uma relação de confiança com o Neptune:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
          "Service": "rds.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

O `RoleA` também teria uma política de permissão que permitisse a ele assumir o `RoleB`, que é de propriedade da Conta B:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1487639602000",
```

```
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": "arn:aws:iam::(Account B ID):role/RoleB"
    }
  ]
}
```

Por outro lado, o `RoleB` teria uma política de confiança para estabelecer uma relação de confiança com o `RoleA`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Principal": {
        "AWS": "arn:aws:iam::(Account A ID):role/RoleA"
      }
    }
  ]
}
```

O `RoleB` também precisaria de permissão para acessar dados no bucket do Amazon S3 localizado na Conta B.

Criação de um AWS Security Token Service endpoint VPC (STS)

O carregador Neptune exige um VPC AWS STS endpoint para quando você está encadeando funções do IAM para acessar de forma privada APIs por meio de endereços IP privados. AWS STS Você pode se conectar diretamente de uma Amazon VPC a AWS STS um VPC Endpoint de forma segura e escalável. Quando você usa um endpoint da VPC de interface, ele fornece um melhor procedimento de segurança, pois você não precisa abrir firewalls de tráfego de saída. Ele também oferece os outros benefícios do uso de endpoints da Amazon VPC.

Ao usar um VPC Endpoint, o tráfego para AWS STS não é transmitido pela Internet e nunca sai da rede Amazon. Sua VPC está conectada com segurança, AWS STS sem riscos de disponibilidade ou restrições de largura de banda no tráfego da rede. Para obter mais informações, consulte Using AWS STS interface VPC endpoints.

Para configurar o acesso para AWS Security Token Service (STS)

1. Faça login AWS Management Console e abra o console da Amazon VPC em https://console.aws.amazon.com/vpc/.
2. No painel de navegação, escolha Endpoints.
3. Escolha Create Endpoint (Criar endpoint).
4. Selecione o Nome do serviço `com.amazonaws.region.sts` para o endpoint do tipo Interface.
5. Selecione a VPC que contém a instância de banco de dados do Neptune e a instância do EC2.
6. Marque a caixa de seleção ao lado da sub-rede na qual a instância do EC2 está presente. Não é possível selecionar várias sub-redes em uma mesma zona de disponibilidade.
7. Em IP address type (Tipo de endereço IP), escolha uma das seguintes opções:
   - IPv4: atribua endereços IPv4 às interfaces de rede de endpoint. Só haverá suporte para esta opção se todas as sub-redes selecionadas tiverem intervalos de endereços IPv4.
   - IPv6: atribua endereços IPv6 às interfaces de rede de endpoint. Essa opção só será compatível se todas as sub-redes selecionadas forem sub-redes somente IPv6.
   - Dualstack: atribua endereços IPv4 e IPv6 às interfaces de rede de endpoint. Só haverá suporte para esta opção se todas as sub-redes selecionadas tiverem intervalos de endereços IPv4 e IPv6.
8. Em Grupos de segurança, selecione os grupos de segurança para associar às interfaces de rede do endpoint para o endpoint da VPC. Você precisaria selecionar todos os grupos de segurança associados à instância de banco de dados do Neptune e à instância do EC2.
9. Em Policy (Política), selecione Full access (Acesso total) para permitir todas as operações de todas as entidades principais em todos os recursos no endpoint da VPC. Ou então selecione Custom (Personalizar) para anexar uma política de endpoint da VPC que controle as permissões das entidades principais para realizar ações em recursos sobre o endpoint da VPC. Essa opção ficará disponível somente se o serviço for compatível com as políticas de endpoint da VPC. Para obter mais informações, consulte Endpoint policies.
10. (Opcional) Para adicionar uma tag, escolha Adicionar nova tag e insira a chave e o valor da tag que você deseja.
11. Escolha Criar endpoint.

Para obter mais informações sobre criação de endpoints, consulte VPC Endpoints no Guia do usuário do Amazon VPC. Observe que o endpoint da VPC do Amazon STS é um pré-requisito obrigatório para o encadeamento de perfis do IAM.

Agora que você concedeu acesso ao AWS STS endpoint, pode se preparar para carregar os dados. Para obter informações sobre os formatos compatíveis, consulte Formatos de dados de carga.

Encadear perfis em um comando de carregador

Você pode especificar o encadeamento de perfis ao executar um comando do carregador incluindo uma lista separada por vírgula de ARNs de perfil no parâmetro. `iamRoleArn`

Embora na maioria das vezes você precise ter apenas dois perfis em uma cadeia, certamente é possível encadear três ou mais. Por exemplo, esse comando do carregador encadeia três perfis:

```
curl -X POST https://localhost:8182/loader \
  -H 'Content-Type: application/json' \
  -d '{
        "source" : "s3://(the target bucket name)/(the target date file name)",
        "iamRoleArn" : "arn:aws:iam::(Account A ID):role/(RoleA),arn:aws:iam::(Account
 B ID):role/(RoleB),arn:aws:iam::(Account C ID):role/(RoleC)",
        "format" : "csv",
        "region" : "us-east-1"
      }'
```

Formatos de dados de carga

A API `Load` do Amazon Neptune é compatível com o carregamento de dados em vários formatos.

Formatos de carregamento de grafos de propriedades

Os dados carregados em um dos seguintes formatos de grafo de propriedades podem então ser consultados usando o Gremlin e o openCypher:

- Formato de dados de carga do Gremlin (`csv`): um formato de valores separados por vírgula (CSV).
- Formato de carregamento de dados openCypher (`opencypher`): um formato de valores separados por vírgula (CSV).

Formatos de carregamento RDF

Para carregar dados do Resource Description Framework (RDF) que você pode consultar usando SPARQL, é possível utilizar um dos seguintes formatos padrão conforme especificado pelo World Wide Web Consortium (W3C):

- N-Triples (`ntriples`) da especificação em https://www.w3.org/TR/n-triples/.
- N-Quads (`nquads`) da especificação em https://www.w3.org/TR/n-quads/.
- RDF/XML (`rdfxml`) da especificação em https://www.w3.org/TR/rdf-syntax-grammar/.
- Turtle (`turtle`) da especificação em https://www.w3.org/TR/turtle/.

Os dados de carregamento devem usar a codificação UTF-8.

> **Important**
>
> Todos os arquivos de dados de carga devem ser codificados em formato UTF-8. Se um arquivo não estiver codificado em UTF-8, o Neptune tentará carregá-lo de qualquer maneira como UTF-8.

Para dados N-Quads e N-triples que incluem caracteres Unicode, sequências de escape `\u`*`xxxxx` são compatíveis. No entanto, o Neptune não é compatível com a normalização. Se houver um valor que exija normalização, ele não corresponderá byte-to-byte durante a consulta. Para obter mais informações sobre normalização, consulte a página Normalização em Unicode.org.

Se os dados não estiverem em um formato compatível, você deverá convertê-los antes de carregá-los.

Uma ferramenta para converter o GraphML para o formato CSV do Neptune está disponível no projeto GraphML2CSV em. GitHub

Suporte à compactação de arquivos de dados de carregamento

O Neptune é compatível com a compactação de arquivos individuais no formato `gzip` ou `bzip2`.

O arquivo compactado deve ter a extensão `.gz` ou `.bz2` e deve ser um único arquivo de texto codificado no formato UTF-8. É possível carregar vários arquivos, mas cada um deve estar em um `.gz`, um `.bz2` ou um texto não compactado separado. Arquivos com extensões, como `.tar`, `.tar.gz` e `.tgz` são compatíveis.

As seções a seguir descrevem os formatos em mais detalhes.

Tópicos

Formato de dados de carga do Gremlin

Para carregar dados do Apache TinkerPop Gremlin usando o formato CSV, você deve especificar os vértices e as bordas em arquivos separados.

O carregador pode carregar vários arquivos de vértice e vários arquivos de ponto em um único trabalho de carga.

Para cada comando de carga, o conjunto de arquivos a serem carregados deve estar na mesma pasta no bucket do Amazon S3 e você deve especificar o nome da pasta para o parâmetro `source`. Os nomes dos arquivos e as extensões dos nomes dos arquivos não são importantes.

O formato CSV do Amazon Neptune segue a especificação RFC 4180 CSV. Para obter mais informações consulte Formato comum e tipo MIME para arquivos CSV no site do Internet Engineering Task Force (IETF).

> **Note**
>
> Todos os arquivos devem ser codificados em formato UTF-8.

Cada arquivo tem uma linha de cabeçalho separada por vírgula. A linha de cabeçalho consiste nos cabeçalhos da coluna do sistema e nos cabeçalhos da coluna de propriedade.

Cabeçalhos de colunas do sistema

Os cabeçalhos da coluna do sistema exigidos e permitidos são diferentes para arquivos de vértice e arquivos de ponto.

Cada coluna do sistema pode aparecer apenas uma vez em um cabeçalho.

Todos os rótulos diferenciam maiúsculas de minúsculas.

Cabeçalhos de vértice

- `~id`: obrigatório

  Um ID para o vértice.
- `~label`

  Um rótulo para o vértice. Vários valores de rótulo são permitidos, separados por ponto e vírgula (`;`).

  Se não `~label` estiver presente, TinkerPop fornece um rótulo com o valor`vertex`, porque cada vértice deve ter pelo menos um rótulo.

Cabeçalhos de ponto

- `~id`: obrigatório

  Um ID para o ponto.
- `~from`: obrigatório

  O ID do vértice do vértice de.
- `~to`: obrigatório

  O ID do vértice do vértice para.
- `~label`

  Um rótulo para o ponto. Os pontos podem ter apenas um único rótulo.

  Se não `~label` estiver presente, TinkerPop fornece uma etiqueta com o valor`edge`, pois cada borda deve ter uma etiqueta.

Cabeçalhos de coluna de propriedade

É possível especificar uma coluna (`:`) para uma propriedade usando a sintaxe a seguir. Os nomes dos tipos não diferenciam maiúsculas de minúsculas. No entanto, observe que, se houver o caractere de dois pontos no nome de uma propriedade, ele deverá ser precedido por uma barra invertida como caractere de escape: `\:`

```
propertyname:type
```

> **Note**
>
> Não são permitidos caracteres de espaço, vírgula, retorno de carro e nova linha nos cabeçalhos das colunas, portanto, os nomes das propriedades não podem incluir esses caracteres.

Você pode especificar uma coluna para um tipo de matriz adicionando `[]` para o tipo:

```
propertyname:type[]
```

> **Note**
>
> As propriedades de borda podem ter apenas um único valor e causarão um erro se um tipo de matriz é especificado ou um segundo valor é especificado.

O exemplo a seguir mostra o cabeçalho da coluna de uma propriedade denominada `age` do tipo `Int`.

```
age:Int
```

Cada linha no arquivo precisa ter um número inteiro nessa posição ou ser deixada em branco.

Matrizes de strings são permitidas, mas as strings em uma matriz não podem incluir o caractere ponto e vírgula (`;`), a menos que ele tenha uma barra invertida como caractere de escape (como este: `\;`).

Especificar a cardinalidade de uma coluna

A partir de Versão 1.0.1.0.200366.0 (26/07/2019), o cabeçalho da coluna pode ser usado para especificar a cardinalidade da propriedade identificada pela coluna. Isso permite que o carregador em massa use a cardinalidade de forma semelhante à forma como as consultas do Gremlin fazem.

Você especifica a cardinalidade de uma coluna da seguinte forma:

```
propertyname:type(cardinality)
```

O valor de *cardinalidade* pode ser `single` ou `set`. O padrão é `set`, o que significa que a coluna pode aceitar vários valores. No caso de arquivos de ponto, a cardinalidade é sempre única e especificar qualquer outra cardinalidade faz com que o carregador gere uma exceção.

Se a cardinalidade for `single`, o carregador lançará um erro se um valor anterior já estiver presente quando um valor for carregado, ou se vários valores forem carregados. Esse comportamento pode ser substituído para que um valor existente seja substituído quando um novo valor é carregado usando o sinalizador `updateSingleCardinalityProperties`. Consulte Comando Loader.

É possível usar uma configuração de cardinalidade com um tipo de matriz, embora isso não seja geralmente necessário. Estas são as combinações possíveis:

- `name:type`: a cardinalidade é `set`, e o conteúdo é um único valor.
- `name:type[]`: a cardinalidade é `set`, e o conteúdo são vários valores.
- `name:type(single)`: a cardinalidade é `single`, e o conteúdo é um único valor.
- `name:type(set)`: a cardinalidade é `set`, equivalente ao padrão, e o conteúdo é um único valor.
- `name:type(set)[]`: a cardinalidade é `set`, e o conteúdo são vários valores.
- `name:type(single)[]`: isso é contraditório e gera um erro.

A seção a seguir lista todos os tipos de dados do Gremlin disponíveis.

Tipos de dados do Gremlin

Esta é uma lista dos tipos de propriedade permitidos, com uma descrição de cada tipo.

Bool (ou booliano)

Indica um campo booliano. Valores permitidos: `false`, `true`

> **Note**
>
> Qualquer valor diferente de `true` será tratado como falso.

Tipos de número inteiro

Os valores fora dos intervalos definidos resultam em um erro.

Tipo	Intervalo

Byte	-128 a 127
Short	-32768 a 32767
Int	-2^31 a 2^31-1
Long	-2^63 a 2^63-1

Tipos de número decimal

Compatíveis com a notação decimal ou com a notação científica. Também permite símbolos, como (+/-) Infinity ou NaN. INF não é compatível.

Tipo	Intervalo
Float	Ponto flutuante de 32 bits IEEE 754
Double	Ponto flutuante de 64 bits IEEE 754

Valores flutuantes e duplos que são muito longos são carregados e arredondados para a precisão do valor mais próximo de 24 bits (flutuante) e 53 bits (duplo). Um valor intermediário é arredondado para zero para os últimos dígitos restantes no nível de bits.

String

Aspas são opcionais. Os caracteres de vírgulas, nova linha e retorno de carro serão escapados automaticamente se forem incluídos em uma sequência entre aspas duplas (`"`). Exemplo: `"Hello, World"`

Para incluir aspas em uma string entre aspas, você pode efetuar o escape as aspas usando duas em seguida: Exemplo: `"Hello ""World"""`

Matrizes de strings são permitidas, mas as strings em uma matriz não podem incluir o caractere ponto e vírgula (`;`), a menos que ele tenha uma barra invertida como caractere de escape (como este: `\;`).

Se desejar usar sequências entre aspas em uma matriz, você deverá colocar a matriz inteira entre um conjunto de aspas. Exemplo: `"String one; String 2; String 3"`

Data

Data do Java no formato ISO-8601. Compatível com os formatos: `yyyy-MM-dd`, `yyyy-MM-ddTHH:mm`, `yyyy-MM-ddTHH:mm:ss`, `yyyy-MM-ddTHH:mm:ssZ`

Formato de linhas do Gremlin

Delimitadores

Os campos em uma linha são separados por uma vírgula. Os registros são separados por uma nova linha ou por uma nova linha seguida por um retorno de carro.

Campos em branco

Os campos em branco são permitidos para colunas não necessárias (como as propriedades definidas pelo usuário). Um campo em branco ainda exige uma vírgula separadora. Campos em branco nas colunas obrigatórias resultarão em um erro de análise. Valores de string vazios são interpretados como valores de string vazios para o campo; não como um campo em branco. O exemplo na próxima seção tem um campo em branco em cada vértice de exemplo.

IDs de vértices

Os valores de `~id` devem ser exclusivos para todos os vértices em cada arquivo de vértice. Várias linhas de vértices com valores idênticos de `~id` são aplicadas a um único vértice no gráfico. A string vazia (`""`) é um id válido e o vértice é criado com uma string vazia como id.

IDs de pontos

Além disso, os valores de `~id` devem ser exclusivos para todos os pontos em cada arquivo de ponto. Várias linhas de pontos com valores idênticos de `~id` são aplicadas ao único ponto no gráfico. A string vazia (`""`) é uma identificação válida e a borda é criada com uma string vazia como a identificação.

Rótulos

As etiquetas diferenciam maiúsculas e minúsculas e não podem estar vazias. Um valor de `""` resultará em um erro.

Valores de sequências

Aspas são opcionais. Os caracteres de vírgulas, nova linha e retorno de carro serão escapados automaticamente se forem incluídos em uma sequência entre aspas duplas (`"`). Valores de string

vazios (`""`) são interpretados como um valor de string vazio para o campo; não como um campo em branco.

Especificação do formato CSV

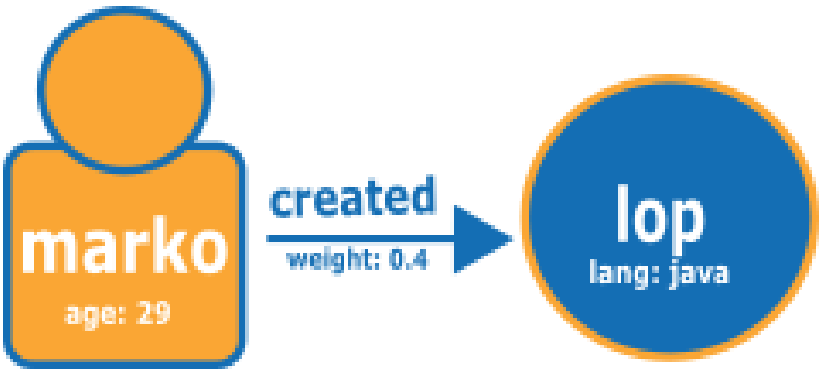
O formato CSV do Neptune segue a especificação RFC 4180 CSV, incluindo os requisitos a seguir.

- Os finais de linha no estilo do Unix e do Windows são compatíveis (\n ou \r\n).
- Qualquer campo pode ser colocado entre aspas (usando aspas duplas).
- Campos que contêm uma quebra de linha, aspas duplas ou vírgulas devem ser colocados entre aspas. (Se não estiverem, o carregamento será abortado imediatamente.)
- Um caractere de aspas duplas (") em um campo deve ser representado por dois caracteres de aspas duplas. Por exemplo, uma sequência `Hello "World"` deve estar presente como `"Hello ""World"""` nos dados.
- Espaços circundantes entre delimitadores são ignorados. Se uma linha estiver presente como`value1, value2`, ela será armazenada como `"value1" "value2"` e.
- Todos os outros caracteres de escape são armazenados textualmente. Por exemplo, `"data1\tdata2"` é armazenado como `"data1\tdata2"`. Nenhum outro escape é necessário, desde que esses caracteres sejam incluídos entre aspas.
- Campos em branco são permitidos. Um campo em branco é considerado um valor vazio.
- Vários valores para um campo são especificados com um ponto e vírgula (`;`) entre os valores.

Para obter mais informações consulte Formato comum e tipo MIME para arquivos CSV no site do Internet Engineering Task Force (IETF).

Exemplo do Gremlin

O diagrama a seguir mostra um exemplo de dois vértices e uma aresta retirados do gráfico TinkerPop moderno.



Veja o grafo no formato de carga do CSV do Neptune.

Arquivo de vértice:

```
~id,name:String,age:Int,lang:String,interests:String[],~label
v1,"marko",29,,"sailing;graphs",person
v2,"lop",,"java",,software
```

Visualização tabular do arquivo de vértice:

~id	name:String	age:Int	lang:String	Interesses: string []	~label
v1	"marko"	29		["navegação", “gráficos"]	pessoa
v2	"lop"		"java"		software

Arquivo de ponto

```
~id,~from,~to,~label,weight:Double
e1,v1,v2,created,0.4
```

Visualização tabular do arquivo de ponto:

~id	~from	~to	~label	weight:Double
e1	v1	v2	created	0.4

Próximos Passos

Agora que você sabe mais sobre os formatos de carregamento, consulte Exemplo: carregamento de dados em uma instância de banco de dados do Neptune.

Formato de carregamento para dados do openCypher

Para carregar dados do openCypher usando o formato CSV do openCypher, é necessário especificar nós e relacionamentos em arquivos separados. O carregador pode carregar vários desses arquivos de nós e arquivos de relacionamento em um único trabalho de carregamento.

Para cada comando de carregamento, o conjunto de arquivos a serem carregados deve ter o mesmo prefixo de caminho em um bucket do Amazon Simple Storage Service. Você especifica esse prefixo no parâmetro de origem. Os nomes e as extensões dos arquivos não são importantes.

No Amazon Neptune, o formato CSV do openCypher está em conformidade com a especificação CSV RFC 4180. Para obter mais informações consulte Common Format and MIME Type for CSV Files (https://tools.ietf.org/html/rfc4180) no site do Internet Engineering Task Force (IETF).

> **Note**
>
> Esses arquivos DEVEM ser codificados em formato UTF-8.

Cada arquivo tem uma linha de cabeçalhos separados por vírgula que contém cabeçalhos de coluna do sistema e cabeçalhos de coluna de propriedades.

Cabeçalhos de coluna do sistema em arquivos de carregamento de dados do openCypher

Uma coluna do sistema específica pode aparecer apenas uma vez em cada arquivo. Todos os rótulos de cabeçalho de coluna do sistema diferenciam maiúsculas de minúsculas.

Os cabeçalhos de coluna do sistema obrigatórios e permitidos são diferentes para arquivos de carregamento de nós e arquivos de carregamento de relacionamento do openCypher:

Cabeçalhos de coluna do sistema em arquivos de nós

- **`:ID`**: (obrigatório) um ID para o nó.

  Um espaço de ID opcional pode ser adicionado ao cabeçalho de coluna `:ID` de nó como este: `:ID(`*`ID Space`*`)`. Um exemplo é `:ID(movies)`.

  Ao carregar relacionamentos que conectem os nós nesse arquivo, use os mesmos espaços de ID nas colunas `:START_ID` e/ou `:END_ID` dos arquivos de relacionamento.

  Também é possível armazenar a coluna `:ID` do nó como uma propriedade no formulário, *`property name`*`:ID`. Um exemplo é `name:ID`.

  Os IDs de nós devem ser exclusivos em todos os arquivos de nós nos carregamentos atuais e anteriores. Se um espaço de ID for usado, os IDs de nós deverão ser exclusivos em todos os arquivos de nós que usam o mesmo espaço de ID nos carregamentos atuais e anteriores.
- **`:LABEL`**: um rótulo para o nó.

Vários valores de rótulo são permitidos, separados por ponto e vírgula (`;`).

Cabeçalhos de coluna do sistema em arquivos de relacionamento

- **`:ID`**: um ID para o relacionamento. Isso é necessário quando `userProvidedEdgeIds` é verdadeiro (o padrão), mas inválido quando `userProvidedEdgeIds` é `false`.

  Os IDs de relacionamento devem ser exclusivos em todos os arquivos de relacionamento nos carregamentos atuais e anteriores.
- **`:START_ID`**: (obrigatório) o ID do nó no qual esse relacionamento começa.

  Opcionalmente, um espaço de ID pode ser associado à coluna de ID inicial no formato `:START_ID(`*`ID Space`*`)`. O espaço de ID atribuído ao ID do nó inicial deve corresponder ao espaço de ID atribuído ao nó no arquivo de nós.
- **`:END_ID`**: (obrigatório) o ID do nó no qual esse relacionamento termina.

  Opcionalmente, um espaço de ID pode ser associado à coluna de ID final no formato `:END_ID(`*`ID Space`*`)`. O espaço de ID atribuído ao ID do nó final deve corresponder ao espaço de ID atribuído ao nó no arquivo de nós.
- **`:TYPE`**: um tipo para o relacionamento. Os relacionamentos só podem ter um único tipo.

> **Note**
>
> Consulte Carregar dados do openCypher para obter informações sobre como os IDs de nós ou relacionamentos duplicados são tratados pelo processo de carregamento em massa.

Cabeçalhos de coluna de propriedades em arquivos de carregamento de dados do openCypher

É possível especificar que uma coluna contenha os valores de uma propriedade específica usando um cabeçalho de coluna de propriedade no seguinte formato:

```
propertyname:type
```

Não são permitidos caracteres de espaço, vírgula, retorno de carro e nova linha nos cabeçalhos das colunas, portanto, os nomes das propriedades não podem incluir esses caracteres. Veja um exemplo de cabeçalho de coluna para uma propriedade chamada `age` do tipo `Int`:

```
age:Int
```

A coluna com `age:Int` como cabeçalho de coluna teria então que conter um valor inteiro ou vazio em cada linha.

Tipos de dados nos arquivos de carregamento de dados do openCypher do Neptune

- **`Bool`** ou **`Boolean`**: um campo booliano. Os valores permitidos são `true` e `false`.

  Qualquer valor diferente de `true` é tratado como `false`.
- **`Byte`**: um número inteiro no intervalo de `-128` a `127`.
- **`Short`**: um número inteiro no intervalo de `-32,768` a `32,767`.
- **`Int`**: um número inteiro no intervalo de `-2^31` a `2^31 - 1`.
- **`Long`**: um número inteiro no intervalo de `-2^63` a `2^63 - 1`.
- **`Float`**: um número de ponto flutuante IEEE 754 de 32 bits. Tanto a notação decimal quanto a notação científica são aceitas. `Infinity`, `-Infinity` e `NaN` são todos reconhecidos, mas `INF` não.

  Valores com muitos dígitos para caberem são arredondados para o valor mais próximo (um valor intermediário é arredondado para 0 para o último dígito restante no nível de bit).
- **`Double`**: um número de ponto flutuante IEEE 754 de 64 bits. Tanto a notação decimal quanto a notação científica são aceitas. `Infinity`, `-Infinity` e `NaN` são todos reconhecidos, mas `INF` não.

  Valores com muitos dígitos para caberem são arredondados para o valor mais próximo (um valor intermediário é arredondado para 0 para o último dígito restante no nível de bit).
- **`String`**: aspas são opcionais. Os caracteres de vírgulas, nova linha e retorno de carro receberão escape automaticamente se forem incluídos em uma string entre aspas duplas (") como `"Hello, World"`.

  Você pode incluir aspas em uma string entre aspas usando duas em uma linha, como `"Hello ""World"""`.
- **`DateTime`**: uma data Java em um dos seguintes formatos ISO-8601:
  - `yyyy-MM-dd`
  - `yyyy-MM-ddTHH:mm`
  - `yyyy-MM-ddTHH:mm:ss`

- `yyyy-MM-ddTHH:mm:ssZ`

Tipos de dados de transmissão automática nos arquivos de carregamento de dados do openCypher do Neptune

Os tipos de dados de transmissão automática são fornecidos para carregar tipos de dados não aceitos nativamente pelo Neptune no momento. Os dados nessas colunas são armazenados como strings, literalmente, sem verificação em relação aos formatos pretendidos. Os seguintes tipos de dados de transmissão automática são permitidos:

- **`Char`**: um campo `Char`. Armazenado como uma string.
- **`Date`**, **`LocalDate`** e **`LocalDateTime`**: consulte Neo4j Temporal Instants para ver uma descrição dos tipos `date`, `localdate` e `localdatetime`. Os valores são carregados literalmente como strings, sem validação.
- **`Duration`**: consulte Neo4j Duration format. Os valores são carregados literalmente como strings, sem validação.
- Ponto: um campo de pontos, para armazenar dados espaciais. Consulte Spatial instants. Os valores são carregados literalmente como strings, sem validação.

Exemplo do formato de carregamento do openCypher

O diagrama a seguir, retirado do gráfico TinkerPop moderno, mostra um exemplo de dois nós e uma relação:

Veja o grafo no formato de carregamento normal do openCypher no Neptune.

Arquivo de nós:

```
:ID,name:String,age:Int,lang:String,:LABEL
v1,"marko",29,,person
v2,"lop",,"java",software
```

Arquivo de relacionamento:

```
:ID,:START_ID,:END_ID,:TYPE,weight:Double
e1,v1,v2,created,0.4
```

Como alternativa, você pode usar espaços de ID e ID como uma propriedade, da seguinte maneira:

Primeiro arquivo de nós:

```
name:ID(person),age:Int,lang:String,:LABEL
"marko",29,,person
```

Segundo arquivo de nós:

```
name:ID(software),age:Int,lang:String,:LABEL
"lop",,"java",software
```

Arquivo de relacionamento:

```
:ID,:START_ID,:END_ID,:TYPE,weight:Double
e1,"marko","lop",created,0.4
```

Formatos de dados de carga do RDF

Para carregar dados do Resource Description Framework (RDF), você pode usar um dos seguintes formatos padrão conforme especificado pelo World Wide Web Consortium (W3C):

- N-Triples (`ntriples`) da especificação em https://www.w3.org/TR/n-triples/
- N-Quads (`nquads`) da especificação em https://www.w3.org/TR/n-quads/
- RDF/XML (`rdfxml`) da especificação em https://www.w3.org/TR/rdf-syntax-grammar/
- Turtle (`turtle`) da especificação em https://www.w3.org/TR/turtle/

> **Important**
>
> Todos os arquivos devem ser codificados em formato UTF-8.

Para dados N-Quads e N-triples que incluem caracteres Unicode, sequências de escape \u*xxxxx* são compatíveis. No entanto, o Neptune não é compatível com a normalização. Se houver um valor que exija normalização, ele não corresponderá byte-to-byte durante a consulta. Para obter mais informações sobre normalização, consulte a página Normalização em Unicode.org.

Próximos Passos

Agora que você sabe mais sobre os formatos de carregamento, consulte Exemplo: carregamento de dados em uma instância de banco de dados do Neptune.

Exemplo: carregamento de dados em uma instância de banco de dados do Neptune

Este exemplo mostra como carregar dados no Amazon Neptune. Salvo indicação em contrário, você deve seguir essas etapas em uma instância do Amazon Elastic Compute Cloud (Amazon EC2) na mesma Amazon Virtual Private Cloud (VPC) que a instância do banco de dados Neptune.

Pré-requisitos para o exemplo de carregamento de dados

Antes de começar, você deve ter o seguinte:

- Uma instância de banco de dados do Neptune.

  Para obter informações sobre como iniciar uma instância de banco de dados do Neptune, consulte Criar um cluster de banco de dados do Neptune.
- Um bucket do Amazon Simple Storage Service (Amazon S3) no qual colocar os arquivos de dados.

  Você pode usar um bucket existente. Se você não tiver um bucket do S3, consulte Create a Bucket no Guia de conceitos básicos do Amazon S3.
- Dados do grafo a serem carregados, em um dos formatos compatíveis com o carregador do Neptune:

  Se você estiver usando o Gremlin para consultar seu gráfico, o Neptune poderá carregar dados no formato comma-separated-values (CSV), conforme descrito em. Formato de dados de carga do Gremlin

Se você estiver usando o openCypher para consultar o grafo, o Neptune também poderá carregar dados em um formato `CSV`específico do openCypher, conforme descrito em Formato de carregamento para dados do openCypher.

Se você estiver usando o SPARQL, o Neptune poderá carregar dados em vários formatos do RDF, conforme descrito em Formatos de dados de carga do RDF.

- Um perfil do IAM para a instância de banco de dados do Neptune assumir que tem uma política do IAM que permite acesso aos arquivos de dados no bucket do S3. A política deve conceder permissões de leitura e lista.

  Para obter informações sobre como criar um perfil que tenha acesso ao Amazon S3 e associá-lo a um cluster do Neptune, consulte Pré-requisitos: perfil do IAM e acesso ao Amazon S3.

  > **Note**
  >
  > A API `Load` do Neptune precisa de acesso de leitura apenas aos arquivos de dados. A política do IAM não precisa permitir acesso de gravação nem acesso a todo o bucket.

- Um endpoint da VPC do Amazon S3. Para obter mais informações, consulte a seção Criar o endpoint da VPC do Amazon S3.

Criar o endpoint da VPC do Amazon S3

O carregador do Neptune requer um endpoint da VPC para Amazon S3.

Para configurar o acesso ao Amazon S3

1. Faça login AWS Management Console e abra o console da Amazon VPC em https://console.aws.amazon.com/vpc/.
2. No painel de navegação à esquerda, escolha Endpoints.
3. Escolha Criar Endpoint.
4. Selecione o Service Name (Nome do serviço) `com.amazonaws.`*`region`*`.s3`.

   > **Note**
   >
   > Se a região aqui estiver incorreta, verifique se a região do console está correta.

5. Selecione a VPC que contém a instância de banco de dados do Neptune.
6. Marque a caixa de seleção ao lado das tabelas de rotas associadas às sub-redes relacionados ao cluster. Se tiver apenas uma tabela de rotas, você deverá selecionar essa caixa.
7. Escolha Criar Endpoint.

Para obter informações sobre criação do endpoint, consulte VPC Endpoints no Guia do usuário da Amazon VPC. Para obter informações sobre as limitações dos endpoints da VPC, consulte VPC Endpoints for Amazon S3.

Como carregar dados em uma instância de banco de dados do Neptune

1. Copie os arquivos de dados em um bucket do Amazon S3. O bucket do S3 deve estar na mesma AWS região do cluster que carrega os dados.

   Você pode usar o AWS CLI comando a seguir para copiar os arquivos para o bucket.

   > **Note**
   >
   > Esse comando não precisa ser executado na instância do Amazon EC2.

   ```
   aws s3 cp data-file-name s3://bucket-name/object-key-name
   ```

   > **Note**
   >
   > No Amazon S3, um nome de chave de objeto é o caminho inteiro de um arquivo, inclusive o nome do arquivo.
   > Exemplo: no comando `aws s3 cp datafile.txt s3://examplebucket/mydirectory/datafile.txt`, o nome da chave do objeto é **`mydirectory/datafile.txt`**.

   Como alternativa, você pode usar o AWS Management Console para fazer upload de arquivos para o bucket do S3. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/ e escolha um bucket. No canto superior esquerdo, selecione Upload (Fazer upload) para fazer upload dos arquivos.

2. Em uma janela de linha de comando, insira o seguinte para executar o carregador do Neptune, usando os valores corretos para o endpoint, o caminho do Amazon S3, o formato e o ARN do perfil do IAM.

   O parâmetro `format` pode ser qualquer um dos seguintes valores: `csv` para Gremlin, `opencypher` para openCypher ou `ntriples`, `nquads`, `turtle` e `rdfxml` para RDF. Para obter informações sobre outros parâmetros, consulte Comando do carregador do Neptune.

   Para obter informações sobre como localizar o nome do host da instância de banco de dados do Neptune, consulte a seção Conectar-se a endpoints do Amazon Neptune.

   O parâmetro da região deve corresponder à região do cluster e do bucket do S3.

   O Amazon Neptune está disponível nas seguintes regiões: AWS

   - Leste dos EUA (Norte da Virgínia): `us-east-1`
   - Leste dos EUA (Ohio): `us-east-2`
   - Oeste dos EUA (N. da Califórnia): `us-west-1`
   - Oeste dos EUA (Oregon): `us-west-2`
   - Canadá (Central): `ca-central-1`
   - América do Sul (São Paulo): `sa-east-1`
   - Europa (Estocolmo): `eu-north-1`
   - Europa (Irlanda): `eu-west-1`
   - Europa (Londres): `eu-west-2`
   - Europa (Paris): `eu-west-3`
   - Europa (Frankfurt): `eu-central-1`
   - Oriente Médio (Bahrein): `me-south-1`
   - Oriente Médio (Emirados Árabes Unidos): `me-central-1`
   - Israel (Tel Aviv): `il-central-1`
   - África (Cidade do Cabo): `af-south-1`
   - Ásia-Pacífico (Hong Kong): `ap-east-1`
   - Ásia-Pacífico (Tóquio): `ap-northeast-1`
   - Ásia-Pacífico (Seul): `ap-northeast-2`
   - Ásia-Pacífico (Osaka): `ap-northeast-3`

- Ásia-Pacífico (Singapura): `ap-southeast-1`
- Ásia-Pacífico (Sydney): `ap-southeast-2`
- Ásia-Pacífico (Mumbai): `ap-south-1`
- China (Pequim): `cn-north-1`
- China (Ningxia): `cn-northwest-1`
- AWS GovCloud (Oeste dos EUA): `us-gov-west-1`
- AWS GovCloud (Leste dos EUA): `us-gov-east-1`

```
curl -X POST \
    -H 'Content-Type: application/json' \
    https://your-neptune-endpoint:port/loader -d '
    {
      "source" : "s3://bucket-name/object-key-name",
      "format" : "format",
      "iamRoleArn" : "arn:aws:iam::account-id:role/role-name",
      "region" : "region",
      "failOnError" : "FALSE",
      "parallelism" : "MEDIUM",
      "updateSingleCardinalityProperties" : "FALSE",
      "queueRequest" : "TRUE",
      "dependencies" : ["load_A_id", "load_B_id"]
    }'
```

Para obter informações sobre como criar e associar um perfil do IAM a um cluster do Neptune, consulte Pré-requisitos: perfil do IAM e acesso ao Amazon S3.

> **Note**
>
> Consulte Parâmetros de solicitação do carregador do Neptune) para obter informações detalhadas sobre parâmetros de solicitação de carga. Resumindo:
> O parâmetro `source` aceita um URI do Amazon S3 que aponte para um único arquivo ou para uma pasta. Se você especificar uma pasta, o Neptune fará upload de cada arquivo de dados na pasta.
> A pasta pode conter vários arquivos de vértice e vários arquivos de ponto.
> O URI pode estar em qualquer um dos seguintes formatos.
>
> - `s3://bucket_name/object-key-name`

- `https://s3.amazonaws.com/bucket_name/object-key-name`
- `https://s3-us-east-1.amazonaws.com/bucket_name/object-key-name`

O parâmetro `format` pode ser:

- Formato CSV do Gremlin (`csv`) para propriedade de grafo do Gremlin
- Formato CSV do openCypher (`opencypher`) para grafos de propriedades do openCypher
- N-Triples (`ntriples`) formato para RDF/SPARQL
- N-Quads (`nquads`) formato para RDF/SPARQL
- RDF/XML (`rdfxml`) formato para RDF/SPARQL
- Turtle (`turtle`) formato para RDF / SPARQL

O parâmetro opcional `parallelism` permite que você restrinja o número de threads usados no processo de carregamento em massa. Ele pode ser definido como `LOW`, `MEDIUM`, `HIGH`, or `OVERSUBSCRIBE`.
Quando `updateSingleCardinalityProperties` estiver definido como `"FALSE"`, o carregador retornará um erro se mais de um valor for fornecido em um arquivo de origem sendo carregado para uma propriedade de vértice de borda ou de cardinalidade única.
Configurar o `queueRequest` para o `"TRUE"` faz com que a solicitação de carga seja colocada em uma fila na qual já existe um trabalho de carga em execução.
O parâmetro `dependencies` torna a execução da solicitação de carga dependente da conclusão com êxito de um ou mais trabalhos de carga que já foram colocados na fila.

3. O carregador do Neptune retorna um trabalho `id` que permite que você confira o status ou cancele o processo de carregamento, por exemplo:

```
{
    "status" : "200 OK",
    "payload" : {
        "loadId" : "ef478d76-d9da-4d94-8ff1-08d9d4863aa5"
    }
}
```

4. Digite o seguinte para obter o status da carga com o `loadId` da Etapa 3:

```
curl -G 'https://your-neptune-endpoint:port/loader/ef478d76-
d9da-4d94-8ff1-08d9d4863aa5'
```

Se o status da carga listar um erro, você poderá solicitar um status mais detalhado e uma lista dos erros. Para ter mais informações e exemplos, consulte API Get-Status do carregador do Neptune.

5. (Opcional) Cancele o trabalho de `Load`.

   Digite o seguinte para `Delete` o trabalho do carregador com o `id` de trabalho da Etapa 3:

   ```
   curl -X DELETE 'https://your-neptune-endpoint:port/loader/ef478d76-
   d9da-4d94-8ff1-08d9d4863aa5'
   ```

   O comando `DELETE` retorna o código HTTP `200 OK` após o cancelamento bem-sucedido.

   Os dados dos arquivos do trabalho de carga que concluíram o carregamento não são revertidos. Os dados permanecem na instância de banco de dados do Neptune.

Otimizar uma carga em massa do Amazon Neptune

Use as seguintes estratégias para reduzir ao mínimo o tempo de carregamento de uma carga em massa do Neptune:

- Limpe os dados:
  - Converta os dados em um formato de dados compatível antes de carregá-los.
  - Remova todas as duplicações ou erros conhecidos.
  - Reduza o número de predicados exclusivos (como propriedades de bordas e vértices) o máximo possível.
- Otimize os arquivos:
  - Se você carregar arquivos grandes, como arquivos CSV, de um bucket do Amazon S3, o carregador gerenciará a simultaneidade para você, analisando-os em partes que poderão ser carregadas paralelamente. Usar um número muito grande de arquivos pequenos pode retardar esse processo.
  - Se você carregar vários arquivos de uma pasta do Amazon S3, o carregador carregará automaticamente os arquivos de vértice primeiro e depois os arquivos de borda.

- A compactação dos arquivos reduz o tempo de transferência. O carregador é compatível com a compactação `gzip` dos arquivos de origem.

- Confira as configurações do carregador:
  - Se você não precisar realizar nenhuma outra operação durante o carregamento, use o parâmetro `OVERSUBSCRIBE` `parallelism`. Essa configuração de parâmetro faz com que o carregador em massa use todos os recursos de CPU disponíveis ao ser executado. Geralmente, são necessários de 60% a 70% de capacidade da CPU para manter a operação funcionando tão rápido quanto permitem as restrições de E/S.

    > **Note**
    >
    > Quando `parallelism` é definido como `OVERSUBSCRIBE` ou `HIGH` (a configuração padrão), há o risco de, ao carregar dados do openCypher, que os threads encontrem uma condição de corrida e um deadlock, gerando um erro `LOAD_DATA_DEADLOCK`. Nesse caso, defina `parallelism` como uma configuração mais baixa e repita o carregamento.

  - Se o trabalho de carregamento incluir várias solicitações de carregamento, use o parâmetro `queueRequest`. A configuração de `queueRequest` como `TRUE` permite que o Neptune coloque suas solicitações na fila para que você não precise esperar que uma termine antes de emitir outra.
  - Se suas solicitações de carregamento estiverem sendo colocadas na fila, você poderá configurar níveis de dependência usando o parâmetro `dependencies`, para que a falha de um trabalho cause falhas nos trabalhos dependentes. Isso pode evitar inconsistências nos dados carregados.
  - Se um trabalho de carregamento envolver a atualização de valores carregados anteriormente, defina o parâmetro `updateSingleCardinalityProperties` como `TRUE`. Caso contrário, o carregador tratará uma tentativa de atualizar um valor de cardinalidade único existente como um erro. No caso de dados do Gremlin, a cardinalidade também é especificada nos cabeçalhos de coluna de propriedades (consulte Cabeçalhos de coluna de propriedade).

    > **Note**
    >
    > O parâmetro `updateSingleCardinalityProperties` não está disponível para dados do Resource Description Framework (RDF).

- É possível usar o parâmetro `failOnError` para determinar se as operações de carregamento em massa devem falhar ou continuar quando um erro for encontrado. Além disso, você pode usar o parâmetro `mode` para garantir que uma tarefa de carregamento seja retomada a partir do ponto em que uma tarefa anterior falhou, em vez de recarregar dados que já haviam sido carregados.
- Aumentar a escala verticalmente: defina a instância de gravador do cluster de banco de dados como o tamanho máximo antes do carregamento em massa. Observe que, se você fizer isso, também deverá aumentar a escala verticalmente de todas as instâncias de réplica de leitura no cluster de banco de dados ou removê-las até terminar de carregar os dados.

  Quando o carregamento em massa estiver concluído, não se esqueça de reduzir a escala verticalmente da instância de gravador novamente.

> **Important**
>
> Se você tiver um ciclo de reinicializações repetidas de réplica de leitura em virtude do atraso na replicação durante o carregamento em massa, é provável que as réplicas não consigam acompanhar o gravador no cluster de banco de dados. Escale os leitores para serem maiores do que o gravador ou remova-os temporariamente durante o carregamento em massa e, depois, recrie-os após a conclusão.

Consulte Parâmetros de solicitação para obter mais detalhes sobre como definir os parâmetros de solicitação do carregador.

Referência do carregador do Neptune

Esta seção descreve as APIs do `Loader` do Amazon Neptune disponíveis no endpoint HTTP de uma instância de banco de dados do Neptune.

> **Note**
>
> Consulte Erro de carregador do Neptune e mensagens de feed para obter uma lista das mensagens de erro e feed geradas pelo carregador em caso de erros.

Sumário

Comando do carregador do Neptune

Carrega dados de um bucket do Amazon S3 em uma instância de banco de dados do Neptune.

Para carregar os dados, você deve enviar uma solicitação HTTP `POST` ao endpoint `https://`*`your-neptune-endpoint`*`:`*`port`*`/loader`. Os parâmetros da solicitação de `loader` podem ser enviados no corpo do `POST` ou como parâmetros codificados por URL.

> **Important**
>
> O tipo MIME deve ser `application/json`.

O bucket do S3 deve estar na mesma AWS região do cluster.

> **Note**
>
> Você poderá carregar dados criptografados do Amazon S3 se ele foi criptografado usando o modo `SSE-S3` do Amazon S3. Nesse caso, o Neptune pode personificar suas credenciais e emitir chamadas `s3:getObject` em seu nome.
> Você também pode carregar dados criptografados do Amazon S3 que foram criptografados usando o modo `SSE-KMS`, desde que o perfil do IAM inclua as permissões necessárias para acessar o AWS KMS. Sem AWS KMS as permissões adequadas, a operação de carregamento em massa falha e retorna uma `LOAD_FAILED` resposta.
> No momento, o Neptune não é compatível com o carregamento de dados criptografados do Amazon S3 usando o modo `SSE-C`.

Não é necessário aguardar a conclusão de um trabalho de carregamento antes de iniciar outro. O Neptune poderá colocar na fila até 64 solicitações de trabalho por vez, desde que os parâmetros `queueRequest` estiverem definidos como `"TRUE"`. A ordem da fila dos trabalhos será first-in-first-out (FIFO). Por outro lado, se não quiser que um trabalho de carga seja colocado na fila, é possível definir o parâmetro `queueRequest` como `"FALSE"` (padrão), para que o trabalho de carga falhe se outro já estiver em andamento.

É possível usar o parâmetro `dependencies` para enfileirar um trabalho que deve ser executado somente após os trabalhos anteriores especificados na fila terem sido concluídos com êxito. Se fizer isso e qualquer um desses trabalhos especificados falharem, o trabalho não será executado e o status será definido como `LOAD_FAILED_BECAUSE_DEPENDENCY_NOT_SATISFIED`.

Sintaxe da solicitação do carregador do Neptune

```
{
  "source" : "string",
  "format" : "string",
  "iamRoleArn" : "string",
  "mode": "NEW|RESUME|AUTO",
  "region" : "us-east-1",
  "failOnError" : "string",
  "parallelism" : "string",
  "parserConfiguration" : {
    "baseUri" : "http://base-uri-string",
    "namedGraphUri" : "http://named-graph-string"
  },
  "updateSingleCardinalityProperties" : "string",
  "queueRequest" : "TRUE",
  "dependencies" : ["load_A_id", "load_B_id"]
}
```

Parâmetros de solicitação do carregador do Neptune

- **`source`**: um URI do Amazon S3.

  O parâmetro SOURCE aceita um URI do Amazon S3 que identifica um único arquivo, vários arquivos, uma pasta ou várias pastas. O Neptune carrega todos os arquivos de dados em qualquer pasta especificada.

  O URI pode estar em qualquer um dos seguintes formatos.

  - `s3://bucket_name/object-key-name`
  - `https://s3.amazonaws.com/bucket_name/object-key-name`
  - `https://s3.us-east-1.amazonaws.com/bucket_name/object-key-name`

  O `object-key-name` elemento do URI é equivalente ao parâmetro de prefixo em uma chamada de API do Amazon ListObjectsS3. Ele identifica todos os objetos no bucket do Amazon S3 especificado cujos nomes começam com esse prefixo. Pode ser um único arquivo ou pasta, ou vários arquivos e/ou pastas.

  A pasta ou as pastas especificadas podem conter vários arquivos de vértice e vários arquivos de borda.

Por exemplo, se você tivesse a seguinte estrutura de pastas e arquivos em um bucket do Amazon S3 chamado: `bucket-name`

```
s3://bucket-name/a/bc
s3://bucket-name/ab/c
s3://bucket-name/ade
s3://bucket-name/bcd
```

Se o parâmetro de origem for especificado como`s3://bucket-name/a`, os três primeiros arquivos serão carregados.

```
s3://bucket-name/a/bc
s3://bucket-name/ab/c
s3://bucket-name/ade
```

- **`format`**: o formato dos dados. Para obter mais informações sobre os formatos de dados para o comando `Loader` do Neptune, consulte Usar o carregador em massa do Amazon Neptune para ingerir dados.

  Valores permitidos

  - **`csv`** para o formato de dados CSV do Gremlin.
  - **`opencypher`** para o formato de dados CSV do openCypher.
  - **`ntriples`** para o formato de dados N-Triples do RDF.
  - **`nquads`** para o formato de dados N-Quads do RDF.
  - **`rdfxml`** para o formato de dados RDF\XML do RDF.
  - **`turtle`** para o formato de dados Turtle do RDF.
- **`iamRoleArn`**: o nome do recurso da Amazon (ARN) de um perfil do IAM a ser assumido pela instância de banco de dados do Neptune para acesso ao bucket do S3. Para obter informações sobre como criar um perfil que tenha acesso ao Amazon S3 e associá-lo a um cluster do Neptune, consulte Pré-requisitos: perfil do IAM e acesso ao Amazon S3.

  A partir da versão 1.2.1.0.R3 do mecanismo, você também pode encadear várias funções do IAM se a instância de banco de dados Neptune e o bucket do Amazon S3 estiverem localizados em contas diferentes. AWS Nesse caso, `iamRoleArn` contém uma lista de ARNs de perfil separados por vírgulas, conforme descrito em Encadear perfis do IAM no Amazon Neptune. Por exemplo: .

```
curl -X POST https://localhost:8182/loader \
  -H 'Content-Type: application/json' \
  -d '{
        "source" : "s3://(the target bucket name)/(the target date file name)",
        "iamRoleArn" : "arn:aws:iam::(Account A
 ID):role/(RoleA),arn:aws:iam::(Account B ID):role/(RoleB),arn:aws:iam::(Account C
 ID):role/(RoleC)",
        "format" : "csv",
        "region" : "us-east-1"
      }'
```

- **`region`**— O `region` parâmetro deve corresponder à AWS região do cluster e ao bucket do S3.

  O Amazon Neptune está disponível nas seguintes regiões da :

  - Leste dos EUA (Norte da Virgínia): `us-east-1`
  - Leste dos EUA (Ohio): `us-east-2`
  - Oeste dos EUA (N. da Califórnia): `us-west-1`
  - Oeste dos EUA (Oregon): `us-west-2`
  - Canadá (Central): `ca-central-1`
  - América do Sul (São Paulo): `sa-east-1`
  - Europa (Estocolmo): `eu-north-1`
  - Europa (Irlanda): `eu-west-1`
  - Europa (Londres): `eu-west-2`
  - Europa (Paris): `eu-west-3`
  - Europa (Frankfurt): `eu-central-1`
  - Oriente Médio (Bahrein): `me-south-1`
  - Oriente Médio (Emirados Árabes Unidos): `me-central-1`
  - Israel (Tel Aviv): `il-central-1`
  - África (Cidade do Cabo): `af-south-1`
  - Ásia-Pacífico (Hong Kong): `ap-east-1`
  - Ásia-Pacífico (Tóquio): `ap-northeast-1`
  - Ásia-Pacífico (Seul): `ap-northeast-2`
  - Ásia-Pacífico (Osaka): `ap-northeast-3`
  - Ásia-Pacífico (Singapura): `ap-southeast-1`

  - Ásia-Pacífico (Sydney): `ap-southeast-2`
  - Ásia-Pacífico (Mumbai): `ap-south-1`
  - China (Pequim): `cn-north-1`
  - China (Ningxia): `cn-northwest-1`
  - AWS GovCloud (Oeste dos EUA): `us-gov-west-1`
  - AWS GovCloud (Leste dos EUA): `us-gov-east-1`
- **`mode`**: o modo do trabalho de carga.

  Valores permitidos: `RESUME`, `NEW`, `AUTO`.

  Valor padrão: `AUTO`

  - `RESUME`: no modo RESUME, o carregador procura uma carga anterior dessa origem e, se encontrar, retomará esse trabalho de carga. Se nenhum trabalho de carga anterior for encontrado, o carregador será interrompido.

    O carregador evita recarregar arquivos que foram carregados com êxito em um trabalho anterior. Ele só tenta processar arquivos com falha. Se você descartou anteriormente dados carregados do cluster do Neptune, esses dados não serão recarregados nesse modo. Se um trabalho de carga anterior carregou todos os arquivos da mesma origem com êxito, nada será recarregado, e o recarregador exibirá êxito.
  - `NEW`: no modo NEW, cria uma solicitação de carga, independentemente de quaisquer cargas anteriores. É possível usar esse modo para recarregar todos dados de uma origem depois de eliminar dados carregados anteriormente do cluster do Neptune ou de carregar novos dados disponíveis na mesma origem.
  - `AUTO`: no modo AUTO, o carregador procura um trabalho de carga anterior da mesma origem e, se encontrar, retomará esse trabalho, assim como no modo `RESUME`.

    Se o carregador não encontrar um trabalho de carga anterior da mesma origem, ele carregará todos os dados da origem, assim como no modo `NEW`.
- **`failOnError`**: sinalizador para alternar uma interrupção completa em um erro.

  Valores permitidos: `"TRUE"`, `"FALSE"`

  Valor padrão: `"TRUE"`.

Quando este parâmetro estiver configurado como `"FALSE"`, o carregador tentará carregar todos os dados no local especificado, ignorando as entradas com erros.

Quando este parâmetro for definido como `"TRUE"`, o carregador será interrompido assim que encontrar um erro. Os dados carregados até esse momento serão mantidos.

- **`parallelism`**: é um parâmetro opcional que pode ser definido para reduzir o número de threads usados pelo processo de carregamento em massa.

  Valores permitidos:

  - `LOW`: o número de threads usados é o número de vCPUs dividido por oito.
  - `MEDIUM`: o número de threads usados é o número de vCPUs disponíveis dividido por dois.
  - `HIGH`: o número de threads usados é equivalente ao número de vCPUs disponíveis.
  - `OVERSUBSCRIBE`: o número de threads usados é o número de vCPUs disponíveis multiplicado por dois. Se esse valor for usado, o carregador em massa usará todos os recursos disponíveis.

    No entanto, isso não significa que a configuração `OVERSUBSCRIBE` resulte em 100% de utilização da CPU. Como a operação de carga está limitada à E/S, a maior utilização esperada da CPU está na faixa de 60% a 70%.

  Valor padrão: `HIGH`

  Às vezes, a configuração `parallelism` pode causar um deadlock entre os threads ao carregar dados do openCypher. Quando isso acontece, o Neptune gera o erro `LOAD_DATA_DEADLOCK`. Geralmente, é possível corrigir o problema definindo uma configuração `parallelism` mais baixa e tentando novamente o comando de carregamento.

- **`parserConfiguration`**: um objeto opcional com valores de configuração de analisador adicional. Os parâmetros filho também são opcionais:

Nome	Valor de exemplo	Descrição
`namedGraphUri`	*`http://aws.amazon.com/neptune/vocab/v01/ DefaultNamedGráfico`*	O gráfico padrão para todos os formatos RDF quando nenhum gráfico for especificado (para formatos non-quads e entradas NQUAD sem gráfico). O padrão é

		`http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`
`baseUri`	*`http://aws.amazon.com/neptune/default`*	O URI base para formatos RDF/XML e Turtle. O padrão é `http://aws.amazon.com/neptune/default` .
`allowEmptyStrings`	*`true`*	É necessário que os usuários do Gremlin possam transmitir valores de string vazios (“”) como propriedades de nós e bordas ao carregar dados CSV. Se `allowEmptyStrings` estiver definido como `false` (o padrão), essas strings vazias serão tratadas como nulas e não serão carregadas. Se `allowEmptyStrings` estiver definido como `true`, o carregador tratará strings vazias como valores de propriedade válidos e as carregará adequadamente.

Para ter mais informações, consulte Gráfico padrão e gráficos nomeados do SPARQL.

- **`updateSingleCardinalityProperties`**: é um parâmetro opcional que controla como o carregador em massa trata um novo valor das propriedades de vértice de cardinalidade única ou de borda. Isso não é compatível com o carregamento de dados do openCypher (consulte Carregar dados do openCypher).

  Valores permitidos: `"TRUE"`, `"FALSE"`

Valor padrão: `"FALSE"`.

Por padrão, ou quando `updateSingleCardinalityProperties` está explicitamente definido como `"FALSE"`, o carregador trata um novo valor como um erro, porque ele viola a cardinalidade única.

Quando `updateSingleCardinalityProperties` está definido como `"TRUE"`, por outro lado, o carregador em massa substitui o valor existente pelo novo. Se vários valores de propriedade de vértice de cardinalidade única ou de ponto forem fornecidos nos arquivos de origem que estão sendo carregados, o valor final quando o carregamento em massa terminar poderá ser qualquer um desses novos valores. O carregador só garante que o valor existente tenha sido substituído por um dos novos.

- **`queueRequest`**: é um parâmetro de sinalizador opcional que indica se a solicitação de carga pode ser colocada na fila ou não.

  Não é necessário esperar um trabalho de carga ser concluído antes de emitir o próximo, porque o Neptune pode colocar na fila até 64 trabalhos por vez, desde que os parâmetros `queueRequest` estejam todos definidos como `"TRUE"`. A ordem da fila dos trabalhos será first-in-first-out (FIFO).

  Se o parâmetro `queueRequest` for omitido ou definido como `"FALSE"`, a solicitação de carga falhará se outro trabalho de carga já estiver em execução.

  Valores permitidos: `"TRUE"`, `"FALSE"`

  Valor padrão: `"FALSE"`.

- **`dependencies`**: é um parâmetro opcional que pode tornar uma solicitação de carga na fila dependente da conclusão com êxito de um ou mais trabalhos anteriores na fila.

  O Neptune poderá colocar na fila até 64 solicitações de carga por vez se os parâmetros `queueRequest` estiverem definidos como `"TRUE"`. O parâmetro `dependencies` permite executar a solicitação enfileirada dependente da conclusão com êxito de uma ou mais solicitações anteriores especificadas na fila.

  Por exemplo, se as cargas `Job-A` e `Job-B` forem interdependentes, mas a carga `Job-C` precisar que a `Job-A` e a `Job-B` sejam concluídas antes de começar, prossiga da seguinte forma:

  1. Envie `load-job-A` e `load-job-B`, uma após a outra, em qualquer ordem e salve os load-ids.
  2. Envie `load-job-C` com os load-ids dos dois trabalhos no campo `dependencies`:

```
"dependencies" : ["job_A_load_id", "job_B_load_id"]
```

Devido ao parâmetro `dependencies`, o carregador em massa não iniciará a `Job-C` até que a `Job-A` e a `Job-B` tenham sido concluídas com êxito. Se uma delas falhar, a Job-C não será executada e o status será definido como `LOAD_FAILED_BECAUSE_DEPENDENCY_NOT_SATISFIED`.

É possível configurar vários níveis de dependência desta forma, para que a falha de um trabalho faça com que todas as solicitações direta ou indiretamente dependentes sejam canceladas.

- **`userProvidedEdgeIds`**: esse parâmetro é necessário somente ao carregar dados do openCypher que contêm IDs de relacionamento. Ele deve ser incluído e definido como `True` quando os IDs de relacionamento do openCypher forem fornecidos explicitamente nos dados de carregamento (recomendado).

  Quando `userProvidedEdgeIds` está ausente ou definido como `True`, uma coluna `:ID` deve estar presente em cada arquivo de relacionamento no carregamento.

  Quando `userProvidedEdgeIds` está presente e definido como `False`, os arquivos de relacionamento no carregamento não devem conter uma coluna `:ID`. Em vez disso, o carregador do Neptune gera automaticamente um ID para cada relacionamento.

  É útil fornecer IDs de relacionamento explicitamente para que o carregador possa retomar o carregamento após a correção do erro nos dados CSV, sem precisar recarregar nenhum relacionamento que já tenha sido carregado. Se os IDs de relacionamento não tiverem sido atribuídos explicitamente, o carregador não poderá retomar um carregamento com falha se for necessário corrigir algum arquivo de relacionamento. Nesse caso, ele deverá carregar todos os relacionamentos.

- `accessKey`: [obsoleto] um ID de chave de acesso de um perfil do IAM com acesso ao bucket do S3 e arquivos de dados.

  Em vez disso, o parâmetro `iamRoleArn` será recomendado. Para obter informações sobre como criar um perfil que tenha acesso ao Amazon S3 e associá-lo a um cluster do Neptune, consulte Pré-requisitos: perfil do IAM e acesso ao Amazon S3.

  Para obter mais informações, consulte Chaves de acesso (ID da chave de acesso e chave de acesso secreta).

- `secretKey`: [obsoleto] O parâmetro `iamRoleArn` será recomendado em vez disso. Para obter informações sobre como criar um perfil que tenha acesso ao Amazon S3 e associá-lo a um cluster do Neptune, consulte Pré-requisitos: perfil do IAM e acesso ao Amazon S3.

  Para obter mais informações, consulte Chaves de acesso (ID da chave de acesso e chave de acesso secreta).

Considerações especiais sobre o carregamento de dados do openCypher

- Ao carregar dados do openCypher no formato CSV, o parâmetro de formato deverá ser definido como `opencypher`.
- O parâmetro `updateSingleCardinalityProperties` não é compatível com carregamento do openCypher porque todas as propriedades do openCypher têm cardinalidade única. O formato de carregamento do openCypher não é compatível com matrizes e, se um valor de ID aparecer mais de uma vez, ele será tratado como uma duplicação ou um erro de inserção (veja abaixo).
- O carregador do Neptune trata as duplicações encontradas nos dados do openCypher da seguinte forma:
  - Se o carregador encontrar várias linhas com o mesmo ID de nó, elas serão mescladas usando a seguinte regra:
    - Todos os rótulos nas linhas são adicionados ao nó.
    - Para cada propriedade, somente um dos valores é carregado. A seleção do que será carregado não é determinística.
  - Se o carregador encontrar várias linhas com o mesmo ID de relacionamento, somente uma delas será carregada. A seleção de uma para carregamento não é determinística.
  - O carregador nunca atualizará os valores das propriedades de um nó ou relacionamento existente no banco de dados se encontrar dados de carregamento com o ID do nó ou do relacionamento existente. No entanto, ele carrega rótulos e propriedades de nós que não estão presentes no nó ou no relacionamento existente.
- Embora você não precise atribuir IDs aos relacionamentos, geralmente é uma boa ideia (veja o parâmetro `userProvidedEdgeIds` acima). Sem IDs de relacionamento explícitos, o carregador deve recarregar todos os relacionamentos em caso de erro em um arquivo de relacionamento, em vez de retomar o carregamento de onde falhou.

  Além disso, se os dados de carregamento não contiverem IDs de relacionamento explícitos, o carregador não terá como detectar relacionamentos duplicados.

Veja a seguir o exemplo de um comando de carregamento do openCypher:

```
curl -X POST https://your-neptune-endpoint:port/loader \
     -H 'Content-Type: application/json' \
     -d '
     {
       "source" : "s3://bucket-name/object-key-name",
       "format" : "opencypher",
       "userProvidedEdgeIds": "TRUE",
       "iamRoleArn" : "arn:aws:iam::account-id:role/role-name",
       "region" : "region",
       "failOnError" : "FALSE",
       "parallelism" : "MEDIUM",
     }'
```

A resposta do carregador é a mesma de sempre. Por exemplo: .

```
{
  "status" : "200 OK",
  "payload" : {
    "loadId" : "guid_as_string"
  }
}
```

Sintaxe da resposta do carregador do Neptune

```
{
    "status" : "200 OK",
    "payload" : {
        "loadId" : "guid_as_string"
    }
}
```

200 OK

O trabalho de carga iniciado com êxito retorna um código `200`.

Erros do carregador do Neptune

Quando ocorre um erro, um objeto JSON é retornado no `BODY` da resposta. O objeto `message` contém uma descrição do erro.

Categorias de erros

- `Error 400`: erros de sintaxe retornam uma solicitação HTTP `400` inadequada. A mensagem descreve o erro.
- `Error 500`: uma solicitação válida que não pode ser processada gera um erro interno do servidor HTTP `500`. A mensagem descreve o erro.

As seguintes são as mensagens de erro possíveis no carregador com uma descrição do erro.

Mensagens de erro do carregador

- `Couldn't find the AWS credential for iam_role_arn` (HTTP 400)

  As credenciais não foram encontradas. Verifique as credenciais fornecidas no console ou na AWS CLI saída do IAM. Assegure-se de ter adicionado o perfil do IAM especificado no `iamRoleArn` ao cluster.
- `S3 bucket not found for source` (HTTP 400)

  O bucket do S3 não existe. Confirme o nome do bucket.
- `The source` *`source-uri`* `does not exist/not reachable` (HTTP 400)

  Nenhum arquivo correspondente foi encontrado no bucket do S3.
- `Unable to connect to S3 endpoint. Provided source =` *`source-uri`* `and region =` *`aws-region`* (HTTP 500)

  Não foi possível se conectar ao Amazon S3. A região deve corresponder à região do cluster. Verifique se você tem um VPC endpoint. Para obter informações sobre como criar um VPC endpoint, consulte Criar o endpoint da VPC do Amazon S3.
- `Bucket is not in provided Region (`*`aws-region`*`)` (HTTP 400)

  O bucket deve estar na mesma AWS região da sua instância de banco de dados Neptune.
- `Unable to perform S3 list operation` (HTTP 400)

  O usuário ou a função do IAM fornecido não tem permissões `List` no bucket ou na pasta. Verifique a política ou a lista de controle de acesso (ACL) no bucket.
- `Start new load operation not permitted on a read replica instance` (HTTP 405)

O carregamento é uma operação de gravação. Repita a carga no endpoint do cluster de leitura/gravação.

- `Failed to start load because of unknown error from S3` (HTTP 500)

  O Amazon S3 gerou um erro desconhecido. Entre em contato com a AWS Support.

- `Invalid S3 access key` (HTTP 400)

  A chave de acesso é inválida. Verifique as credenciais fornecidas.

- `Invalid S3 secret key` (HTTP 400)

  A chave secreta é inválida. Verifique as credenciais fornecidas.

- `Max concurrent load limit breached` (HTTP 400)

  Se uma solicitação de carga for enviada sem `"queueRequest" : "TRUE"` e um trabalho de carga estiver atualmente em execução, a solicitação falhará com esse erro.

- `Failed to start new load for the source "`*`source name`*`". Max load task queue size limit breached. Limit is 64` (HTTP 400)

  O Neptune é compatível com a colocação de até 64 trabalhos de carregador na fila por vez. Se uma solicitação de carga adicional for enviada para a fila quando esta já contiver 64 trabalhos, a solicitação falhará com essa mensagem.

Exemplos do carregador do Neptune

Example Solicitação

O seguinte é uma solicitação enviada por HTTP POST usando o comando `curl`. Ele carrega um arquivo no formato CSV do Neptune. Para ter mais informações, consulte Formato de dados de carga do Gremlin.

```
curl -X POST \
    -H 'Content-Type: application/json' \
    https://your-neptune-endpoint:port/loader -d '
    {
      "source" : "s3://bucket-name/object-key-name",
      "format" : "csv",
      "iamRoleArn" : "ARN for the IAM role you are using",
      "region" : "region",
```

```
      "failOnError" : "FALSE",
      "parallelism" : "MEDIUM",
      "updateSingleCardinalityProperties" : "FALSE",
      "queueRequest" : "FALSE"
    }'
```

Example Resposta

```
{
    "status" : "200 OK",
    "payload" : {
        "loadId" : "ef478d76-d9da-4d94-8ff1-08d9d4863aa5"
    }
}
```

API Get-Status do carregador do Neptune

Obtém o status de trabalho do `loader`.

Para obter o status da carga, você deve enviar uma solicitação HTTP `GET` ao endpoint `https://your-neptune-endpoint:port/loader`. Para obter o status de uma determinada solicitação de carga, você deve incluir o `loadId` como um parâmetro da URL ou acrescentar `loadId` ao caminho da URL.

O Neptune mantém o controle somente dos 1.024 trabalhos de carregamento em massa mais recentes e armazena somente os últimos dez mil detalhes de erro por trabalho.

Consulte Erro de carregador do Neptune e mensagens de feed para obter uma lista das mensagens de erro e feed geradas pelo carregador em caso de erros.

Sumário

Solicitações de Get-Status do carregador do Neptune

Sintaxe da solicitação Get-Status do carregador

```
GET https://your-neptune-endpoint:port/loader?loadId=loadId
```

```
GET https://your-neptune-endpoint:port/loader/loadId
```

```
GET https://your-neptune-endpoint:port/loader
```

Parâmetros da solicitação Get-Status do carregador do Neptune

- **`loadId`**: o ID do trabalho de carga. Se você não especificar um `loadId`, uma lista de IDs de carga será retornada.
- **`details`**: inclua detalhes além do status geral.

  Valores permitidos: `TRUE`, `FALSE`

  Valor padrão: `FALSE`.
- **`errors`**: inclua a lista de erros.

  Valores permitidos: `TRUE`, `FALSE`

  Valor padrão: `FALSE`.

  A lista de erros é paginada. Os parâmetros `page` e `errorsPerPage` permitem que você pagine por todos os erros.
- **`page`**: o número da página do erro. Válido apenas com o parâmetro `errors` definido como `TRUE`.

Valores permitidos: inteiros positivos.

Valor padrão: 1.

- **`errorsPerPage`**: o número de erros por página. Válido apenas com o parâmetro `errors` definido como `TRUE`.

  Valores permitidos: inteiros positivos.

  Valor padrão: 10
- **`limit`**: o número de IDs de cargas a serem listados. Válido somente ao solicitar uma lista de IDs de carga enviando uma solicitação `GET` sem o `loadId` especificado.

  Valores permitidos: inteiros positivos de 1 a 100.

  Valor padrão: 100
- **`includeQueuedLoads`**: um parâmetro opcional que pode ser usado para excluir os IDs de carga de solicitações de carga em fila quando uma lista de IDs de carga for solicitada.

  > **Note**
  >
  > Esse parâmetro está disponível a partir da versão 1.0.3.0 do mecanismo do Neptune.

  Por padrão, os IDs de carga de todos os trabalhos de carga com status `LOAD_IN_QUEUE` estão incluídos nessa lista. Eles aparecem antes dos IDs de outros trabalhos, classificados de acordo com o momento em que foram adicionados à fila, do mais recente para o mais antigo.

  Valores permitidos: `TRUE`, `FALSE`

  Valor padrão: `TRUE`.

Respostas Get-Status do carregador do Neptune

Layout JSON de respostas Get-Status do carregador do Neptune

O layout geral de uma resposta de status do carregador é o seguinte:

```
{
    "status" : "200 OK",
```

```
    "payload" : {
        "feedCount" : [
            {
                "LOAD_FAILED" : number
            }
        ],
        "overallStatus" : {
            "fullUri" : "s3://bucket/key",
            "runNumber" : number,
            "retryNumber" : number,
            "status" : "string",
            "totalTimeSpent" : number,
            "startTime" : number,
            "totalRecords" : number,
            "totalDuplicates" : number,
            "parsingErrors" : number,
            "datatypeMismatchErrors" : number,
            "insertErrors" : number,
        },
        "failedFeeds" : [
            {
                "fullUri" : "s3://bucket/key",
                "runNumber" : number,
                "retryNumber" : number,
                "status" : "string",
                "totalTimeSpent" : number,
                "startTime" : number,
                "totalRecords" : number,
                "totalDuplicates" : number,
                "parsingErrors" : number,
                "datatypeMismatchErrors" : number,
                "insertErrors" : number,
            }
        ],
        "errors" : {
            "startIndex" : number,
            "endIndex" : number,
            "loadId" : "string,
            "errorLogs" : [ ]
        }
    }
}
```

Objetos de resposta **`overallStatus`** e **`failedFeeds`** Get-Status do carregador do Neptune

As possíveis respostas exibidas para cada feed com falha, incluindo as descrições dos erros, são as mesmas para o objeto `overallStatus` em uma resposta `Get-Status`.

Os seguintes campos são exibidos no objeto `overallStatus` de todas as cargas e o objeto `failedFeeds` de cada feed com falha:

- **`fullUri`** : o URI do arquivo ou dos arquivos a serem carregados.

  Tipo: string

  Formato: `s3://`*`bucket`*`/`*`key`*.
- **`runNumber`**: o número de execução dessa carga ou feed. É incrementado quando a carga é reiniciada.

  Tipo: longo não assinado.
- **`retryNumber`**: o número de repetição dessa carga ou feed. É incrementado quando o carregador tenta novamente um feed ou carga automaticamente.

  Tipo: longo não assinado.
- **`status`**: o status exibido da carga ou do feed. `LOAD_COMPLETED` indica uma carga bem-sucedida sem problemas. Para obter uma lista de outras mensagens de status de carregamento, consulte Erro de carregador do Neptune e mensagens de feed.

  Tipo: string.
- **`totalTimeSpent`**: o tempo, em segundos, gasto para analisar e inserir os dados para a carga ou o feed. Isso não inclui o tempo gasto para buscar a lista de arquivos de origem.

  Tipo: longo não assinado.
- **`totalRecords`**: total de registros carregados ou que tentaram carregar.

  Tipo: longo não assinado.

  Observe que, ao realizar o carregamento de um arquivo CSV, a contagem de registros não se refere ao número de linhas carregadas, mas sim ao número de registros individuais nessas linhas. Por exemplo, pegue como exemplo um pequeno arquivo CSV como este:

```
~id,~label,name,team
```

```
'P-1','Player','Stokes','England'
```

O Neptune consideraria que esse arquivo contém três registros, a saber:

```
P-1  label Player
P-1  name  Stokes
P-1  team  England
```

- **`totalDuplicates`**: o número de registros duplicados encontrados.

  Tipo: longo não assinado.

  Como no caso da contagem `totalRecords`, esse valor contém o número de registros duplicados individuais em um arquivo CSV, não o número de linhas duplicadas. Pegue como exemplo este pequeno arquivo CSV:

```
~id,~label,name,team
P-2,Player,Kohli,India
P-2,Player,Kohli,India
```

  O status exibido após o carregamento ficaria assim, relatando seis registros no total, dos quais três são duplicados:

```
{
  "status": "200 OK",
  "payload": {
    "feedCount": [
      {
        "LOAD_COMPLETED": 1
      }
    ],
    "overallStatus": {
      "fullUri": "(the URI of the CSV file)",
      "runNumber": 1,
      "retryNumber": 0,
      "status": "LOAD_COMPLETED",
      "totalTimeSpent": 3,
      "startTime": 1662131463,
      "totalRecords": 6,
      "totalDuplicates": 3,
      "parsingErrors": 0,
      "datatypeMismatchErrors": 0,
```

```
      "insertErrors": 0
    }
  }
}
```

Para carregamento do openCypher, uma duplicação é contada quando:

- O carregador detecta que uma linha em um arquivo de nó tem um ID sem um espaço de ID que é igual a outro valor de ID sem um espaço de ID, seja em outra linha ou pertencente a um nó existente.
- O carregador detecta que uma linha em um arquivo de nó tem um ID com um espaço de ID igual a outro valor de ID com um espaço de ID, seja em outra linha ou pertencente a um nó existente.

Consulte Considerações especiais sobre o carregamento de dados do openCypher.

- **`parsingErrors`**: o número de erros de análise encontrados.

  Tipo: longo não assinado.
- **`datatypeMismatchErrors`**: o número de registros com um tipo de dados que não corresponderam aos dados fornecidos.

  Tipo: longo não assinado.
- **`insertErrors`**: o número de registros que não puderam ser inseridos devido a erros.

  Tipo: longo não assinado.

Objetos de resposta **`errors`** Get-Status do carregador do Neptune

Os erros se enquadram nas seguintes categorias:

- **`Error 400`**: um `loadId` inválido gera um erro de solicitação HTTP `400` inadequada. A mensagem descreve o erro.
- **`Error 500`**: uma solicitação válida que não pode ser processada gera um erro interno do servidor HTTP `500`. A mensagem descreve o erro.

Consulte Erro de carregador do Neptune e mensagens de feed para obter uma lista das mensagens de erro e feed geradas pelo carregador em caso de erros.

Quando ocorre um erro, um objeto JSON `errors` é gerado no `BODY` da resposta, com os seguintes campos:

- **`startIndex`**: o índice do primeiro erro incluído.

  Tipo: longo não assinado.

- **`endIndex`**: o índice do último erro incluído.

  Tipo: longo não assinado.

- **`loadId`**: o ID da carga. Você pode usar esse ID para imprimir os erros do carregamento definindo o parâmetro `errors` como `TRUE`.

  Tipo: string.

- **`errorLogs`**: uma lista dos erros.

  Tipo: lista.

Objetos de resposta **`errorLogs`** Get-Status do carregador do Neptune

O objeto `errorLogs` em `errors` na resposta Get-Status do carregador contém um objeto que descreve cada erro usando os seguintes campos:

- **`errorCode`**: identifica a natureza do erro.

  Pode utilizar um dos seguintes valores:

  - `PARSING_ERROR`
  - `S3_ACCESS_DENIED_ERROR`
  - `FROM_OR_TO_VERTEX_ARE_MISSING`
  - `ID_ASSIGNED_TO_MULTIPLE_EDGES`
  - `SINGLE_CARDINALITY_VIOLATION`
  - `FILE_MODIFICATION_OR_DELETION_ERROR`
  - `OUT_OF_MEMORY_ERROR`
  - `INTERNAL_ERROR` (gerado quando o carregador em massa não consegue determinar o tipo do erro).

- **`errorMessage`**: uma mensagem descrevendo o erro.

  Pode ser uma mensagem genérica associada ao código de erro ou uma mensagem específica que contenha detalhes, por exemplo, sobre um vértice de/para ausente ou sobre um erro de análise.

- **`fileName`**: o nome do feed.

- **`recordNum`**: no caso de um erro de análise, esse é o número do registro no arquivo do registro que não pôde ser analisado. Será definido como zero se o número do registro não for aplicável ao erro ou se não puder ser determinado.

Por exemplo, o carregador em massa geraria um erro de análise se encontrasse uma linha com defeito, como a seguinte, em um arquivo RDF `nquads`:

```
<http://base#subject> |http://base#predicate> <http://base#true> .
```

Como você pode ver, o segundo `http` na linha acima deve ser precedido por < em vez de `|`. O objeto de erro resultante abaixo de `errorLogs` em uma resposta de status deverá ser semelhante a este:

```
{
    "errorCode" : "PARSING_ERROR",
    "errorMessage" : "Expected '<', found: |",
    "fileName" : "s3://bucket/key",
    "recordNum" : 12345
},
```

Exemplos de Get-Status do carregador do Neptune

Exemplo de solicitação do status da carga

O seguinte é uma solicitação enviada por HTTP `GET` usando o comando `curl`.

```
curl -X GET 'https://your-neptune-endpoint:port/loader/loadId (a UUID)'
```

Example Resposta

```
{
    "status" : "200 OK",
    "payload" : {
        "feedCount" : [
            {
                "LOAD_FAILED" : 1
            }
        ],
        "overallStatus" : {
            "datatypeMismatchErrors" : 0,
```

```
            "fullUri" : "s3://bucket/key",
            "insertErrors" : 0,
            "parsingErrors" : 5,
            "retryNumber" : 0,
            "runNumber" : 1,
            "status" : "LOAD_FAILED",
            "totalDuplicates" : 0,
            "totalRecords" : 5,
            "totalTimeSpent" : 3.0
        }
    }
}
```

Exemplo de solicitação de loadIs

O seguinte é uma solicitação enviada por HTTP `GET` usando o comando `curl`.

```
curl -X GET 'https://your-neptune-endpoint:port/loader?limit=3'
```

Example Resposta

```
{
    "status" : "200 OK",
    "payload" : {
         "loadIds" : [
            "a2c0ce44-a44b-4517-8cd4-1dc144a8e5b5",
            "09683a01-6f37-4774-bb1b-5620d87f1931",
            "58085eb8-ceb4-4029-a3dc-3840969826b9"
        ]
    }
}
```

Exemplo de solicitação de status detalhado

O seguinte é uma solicitação enviada por HTTP `GET` usando o comando `curl`.

```
curl -X GET 'https://your-neptune-endpoint:port/loader/loadId (a UUID)?details=true'
```

Example Resposta

```
{
    "status" : "200 OK",
```

```
    "payload" : {
        "failedFeeds" : [
            {
                "datatypeMismatchErrors" : 0,
                "fullUri" : "s3://bucket/key",
                "insertErrors" : 0,
                "parsingErrors" : 5,
                "retryNumber" : 0,
                "runNumber" : 1,
                "status" : "LOAD_FAILED",
                "totalDuplicates" : 0,
                "totalRecords" : 5,
                "totalTimeSpent" : 3.0
            }
        ],
        "feedCount" : [
            {
                "LOAD_FAILED" : 1
            }
        ],
        "overallStatus" : {
            "datatypeMismatchErrors" : 0,
            "fullUri" : "s3://bucket/key",
            "insertErrors" : 0,
            "parsingErrors" : 5,
            "retryNumber" : 0,
            "runNumber" : 1,
            "status" : "LOAD_FAILED",
            "totalDuplicates" : 0,
            "totalRecords" : 5,
            "totalTimeSpent" : 3.0
        }
    }
}
```

Exemplos de Get-Status **`errorLogs`** do carregador do Neptune

Exemplo de resposta de status detalhada quando ocorreram erros

Trata-se de uma solicitação enviada via HTTP `GET` usando `curl`:

```
curl -X GET 'https://your-neptune-endpoint:port/loader/0a237328-afd5-4574-a0bc-c29ce5f54802?details=true&errors=true&page=1&errorsPerPage=3'
```

Example de uma resposta detalhada quando ocorreram erros

Este é um exemplo da resposta que você pode obter da consulta acima, com um objeto `errorLogs` listando os erros de carregamento encontrados:

```
{
    "status" : "200 OK",
    "payload" : {
        "failedFeeds" : [
            {
                "datatypeMismatchErrors" : 0,
                "fullUri" : "s3://bucket/key",
                "insertErrors" : 0,
                "parsingErrors" : 5,
                "retryNumber" : 0,
                "runNumber" : 1,
                "status" : "LOAD_FAILED",
                "totalDuplicates" : 0,
                "totalRecords" : 5,
                "totalTimeSpent" : 3.0
            }
        ],
        "feedCount" : [
            {
                "LOAD_FAILED" : 1
            }
        ],
        "overallStatus" : {
            "datatypeMismatchErrors" : 0,
            "fullUri" : "s3://bucket/key",
            "insertErrors" : 0,
            "parsingErrors" : 5,
            "retryNumber" : 0,
            "runNumber" : 1,
            "status" : "LOAD_FAILED",
            "totalDuplicates" : 0,
            "totalRecords" : 5,
            "totalTimeSpent" : 3.0
        },
        "errors" : {
            "endIndex" : 3,
            "errorLogs" : [
                {
                    "errorCode" : "PARSING_ERROR",
```

```
                    "errorMessage" : "Expected '<', found: |",
                    "fileName" : "s3://bucket/key",
                    "recordNum" : 1
                },
                {
                    "errorCode" : "PARSING_ERROR",
                    "errorMessage" : "Expected '<', found: |",
                    "fileName" : "s3://bucket/key",
                    "recordNum" : 2
                },
                {
                    "errorCode" : "PARSING_ERROR",
                    "errorMessage" : "Expected '<', found: |",
                    "fileName" : "s3://bucket/key",
                    "recordNum" : 3
                }
            ],
            "loadId" : "0a237328-afd5-4574-a0bc-c29ce5f548002",
            "startIndex" : 1
        }
    }
}
```

Exemplo de erro **`Data prefetch task interrupted`**

Ocasionalmente, quando você obtém um status `LOAD_FAILED` e solicita informações mais detalhadas, o erro retornado pode ser um `PARSING_ERROR` com uma mensagem `Data prefetch task interrupted`, da seguinte forma:

```
"errorLogs" : [
    {
        "errorCode" : "PARSING_ERROR",
        "errorMessage" : "Data prefetch task interrupted: Data prefetch task for 11467
 failed",
        "fileName" : "s3://some-source-bucket/some-source-file",
        "recordNum" : 0
    }
]
```

Esse erro ocorre quando há uma interrupção temporária no processo de carregamento de dados que normalmente não foi causada por sua solicitação ou seus dados. De maneira geral, ele pode ser resolvido simplesmente executando a solicitação de upload em massa novamente. Se você

estiver usando as configurações padrão, ou seja, `"mode":"AUTO"` e `"failOnError":"TRUE"`, o carregador ignorará os arquivos que ele já tiver carregado com êxito e retomará o carregamento de arquivos que ainda não tinha carregado quando a interrupção ocorreu.

Trabalho de cancelamento do carregador do Neptune

Cancela um trabalho de carga.

Para cancelar um trabalho, você deve enviar uma solicitação HTTP `DELETE` ao endpoint `https://`*`your-neptune-endpoint`*`:`*`port`*`/loader`. O `loadId` pode ser acrescentado ao caminho da URL `/loader` ou incluído como uma variável na URL.

Sintaxe da solicitação de trabalho de cancelamento

```
DELETE https://your-neptune-endpoint:port/loader?loadId=loadId
```

```
DELETE https://your-neptune-endpoint:port/loader/loadId
```

Parâmetros da solicitação de trabalho de cancelamento

loadId

O ID do trabalho de carga.

Sintaxe da resposta de trabalho de cancelamento

```
no response body
```

200 OK

O trabalho de carga excluído com êxito retorna um código `200`.

Erros do trabalho de cancelamento

Quando ocorre um erro, um objeto JSON é retornado no `BODY` da resposta. O objeto `message` contém uma descrição do erro.

Categorias de erros

- **`Error 400`**: um `loadId` inválido gera um erro de solicitação HTTP `400` inadequada. A mensagem descreve o erro.

- **`Error 500`**: uma solicitação válida que não pode ser processada gera um erro interno do servidor HTTP `500`. A mensagem descreve o erro.

Mensagens de erro do trabalho de cancelamento

As seguintes são as mensagens de erro possíveis da API de cancelamento com uma descrição do erro.

- `The load with id = load_id does not exist or not active` (HTTP 404): a carga não foi encontrada. Verifique o valor do parâmetro `id`.
- `Load cancellation is not permitted on a read replica instance.` (HTTP 405): o carregamento é uma operação de gravação. Repita a carga no endpoint do cluster de leitura/gravação.

Exemplos do trabalho de cancelamento

Example Solicitação

O seguinte é uma solicitação enviada por HTTP `DELETE` usando o comando `curl`.

```
curl -X DELETE 'https://your-neptune-endpoint:port/loader/0a237328-afd5-4574-a0bc-c29ce5f54802'
```

Usando AWS Database Migration Service para carregar dados no Amazon Neptune a partir de um armazenamento de dados diferente

AWS Database Migration Service (AWS DMS) pode carregar dados no Neptune a partir de bancos de dados de origem compatíveis de forma rápida e segura. O banco de dados de origem permanece totalmente operacional durante a migração, o que minimiza o tempo de inatividade de aplicativos que dependem dele.

Você pode encontrar informações detalhadas sobre isso AWS DMS no Guia do AWS Database Migration Service usuário e na Referência AWS Database Migration Service da API. Em específico, é possível descobrir como configurar um cluster do Neptune como destino para migração em Using Amazon Neptune as a Target for AWS Database Migration Service.

Veja alguns pré-requisitos para importar dados para o Neptune usando o AWS DMS:

- Você precisará criar um objeto de mapeamento de AWS DMS tabela para definir como os dados devem ser extraídos do banco de dados de origem (consulte Especificação da seleção e transformações de tabelas por mapeamento de tabelas usando JSON no Guia do AWS DMS usuário para obter detalhes). Esse objeto de configuração de mapeamento de tabelas especifica quais tabelas devem ser lidas e em que ordem e como suas colunas são chamadas. Ele também pode filtrar as linhas que estão sendo copiadas e fornecer transformações de valor simples, como converter para letras minúsculas ou arredondar.
- Será necessário criar um `GraphMappingConfig` do Neptune para especificar como os dados extraídos do banco de dados de origem devem ser carregados no Neptune. Para dados do RDF (consultados usando o SPARQL), o `GraphMappingConfig` é escrito na linguagem de mapeamento R2RML padrão do W3. Para dados de grafos de propriedades (consultados usando o Gremlin), o `GraphMappingConfig` é um objeto JSON, descrito em GraphMappingConfig Layout para dados de gráfico de propriedade/Gremlin.
- Você deve usar AWS DMS para criar uma instância de replicação na mesma VPC do seu cluster de banco de dados Neptune, para mediar a transferência de dados.
- Também será necessário um bucket do Amazon S3 a ser usado como armazenamento intermediário para preparar os dados de migração.

Criando um Netuno GraphMappingConfig

O `GraphMappingConfig` criado especifica como os dados extraídos de um armazenamento de dados de origem devem ser carregados em um cluster de banco de dados do Neptune. O formato será diferente dependendo se ele for destinado a carregar dados do RDF ou a carregar dados de grafos de propriedades.

Para dados do RDF, é possível usar a linguagem R2RML do W3 para mapear dados relacionais para o RDF.

Se você estiver carregando dados de gráficos de propriedades a serem consultados usando o Gremlin, crie um objeto JSON para `GraphMappingConfig`.

GraphMappingConfig Layout para dados RDF/SPARQL

Se você estiver carregando dados do RDF a serem consultados usando o SPARQL, escreva o `GraphMappingConfig` em R2RML. `R2RML` é uma linguagem W3 padrão para mapear dados relacionais para o RDF. Veja a seguir um exemplo:

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix ex: <http://example.com/ns#> .

<#TriplesMap1>
    rr:logicalTable [ rr:tableName "nodes" ];
    rr:subjectMap [
        rr:template "http://data.example.com/employee/{id}";
        rr:class ex:Employee;
    ];
    rr:predicateObjectMap [
        rr:predicate ex:name;
        rr:objectMap [ rr:column "label" ];
    ] .
```

Aqui está outro exemplo:

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix ex: <http://example.com/#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<#TriplesMap2>
    rr:logicalTable [ rr:tableName "Student" ];
    rr:subjectMap   [ rr:template "http://example.com/{ID}{Name}";
                      rr:class foaf:Person ];
    rr:predicateObjectMap [
        rr:predicate ex:id ;
        rr:objectMap  [ rr:column "ID";
                        rr:datatype xsd:integer ]
    ];
    rr:predicateObjectMap [
        rr:predicate foaf:name ;
        rr:objectMap  [ rr:column "Name" ]
    ] .
```

A recomendação do W3 em R2RML: RDB to RDF Mapping Language fornece detalhes da linguagem.

GraphMappingConfig Layout para dados de gráfico de propriedade/Gremlin

Um `GraphMappingConfig` comparável para dados de gráficos de propriedades é um objeto JSON que fornece uma regra de mapeamento para cada entidade de gráfico a ser gerada com base nos dados de origem. O modelo a seguir mostra como é cada regra nesse objeto:

```
{
  "rules": [
    {
      "rule_id": "(an identifier for this rule)",
      "rule_name": "(a name for this rule)",
      "table_name": "(the name of the table or view being loaded)",
      "vertex_definitions": [
        {
          "vertex_id_template": "{col1}",
          "vertex_label": "(the vertex to create)",
          "vertex_definition_id": "(an identifier for this vertex)",
          "vertex_properties": [
            {
              "property_name": "(name of the property)",
              "property_value_template": "{col2} or text",
              "property_value_type": "(data type of the property)"
            }
          ]
        }
      ]
    },
    {
      "rule_id": "(an identifier for this rule)",
      "rule_name": "(a name for this rule)",
      "table_name": "(the name of the table or view being loaded)",
      "edge_definitions": [
        {
          "from_vertex": {
            "vertex_id_template": "{col1}",
            "vertex_definition_id": "(an identifier for the vertex referenced above)"
          },
          "to_vertex": {
            "vertex_id_template": "{col3}",
            "vertex_definition_id": "(an identifier for the vertex referenced above)"
          },
          "edge_id_template": {
            "label": "(the edge label to add)",
```

```
            "template": "{col1}_{col3}"
          },
          "edge_properties":[
            {
              "property_name": "(the property to add)",
              "property_value_template": "{col4} or text",
              "property_value_type": "(data type like String, int, double)"
            }
          ]
        }
      ]
    }
  ]
}
```

Observe que a presença de um rótulo de vértice implica que o vértice está sendo criado aqui, enquanto sua ausência implica que o vértice foi criado por uma fonte diferente e que esta definição está apenas adicionando propriedades do vértice.

Veja a seguir um exemplo de regra para um registro de funcionário:

```
{
  "rules": [
    {
      "rule_id": "1",
      "rule_name": "vertex_mapping_rule_from_nodes",
      "table_name": "nodes",
      "vertex_definitions": [
        {
          "vertex_id_template": "{emp_id}",
          "vertex_label": "employee",
          "vertex_definition_id": "1",
          "vertex_properties": [
            {
              "property_name": "name",
              "property_value_template": "{emp_name}",
              "property_value_type": "String"
            }
          ]
        }
      ]
    },
    {
```

```
      "rule_id": "2",
      "rule_name": "edge_mapping_rule_from_emp",
      "table_name": "nodes",
      "edge_definitions": [
        {
          "from_vertex": {
            "vertex_id_template": "{emp_id}",
            "vertex_definition_id": "1"
          },
          "to_vertex": {
            "vertex_id_template": "{mgr_id}",
            "vertex_definition_id": "1"
          },
          "edge_id_template": {
            "label": "reportsTo",
            "template": "{emp_id}_{mgr_id}"
          },
          "edge_properties":[
            {
              "property_name": "team",
              "property_value_template": "{team}",
              "property_value_type": "String"
            }
          ]
        }
      ]
    }
  ]
}
```

Criando uma tarefa AWS DMS de replicação com Neptune como destino

Depois de criar as configurações de mapeamento de tabelas e mapeamento de grafos, use o processo a seguir para carregar dados do armazenamento de origem no Neptune. Consulte a AWS DMS documentação para obter mais detalhes sobre as APIs em questão.

Etapa 1: criar uma instância de AWS DMS replicação

Crie uma instância de AWS DMS replicação na VPC em que seu cluster de banco de dados Neptune está sendo executado (consulte Como trabalhar com AWS uma instância e instância de replicação do DMS no Guia do usuário). CreateReplication AWS DMS Você pode usar um AWS CLI comando como o seguinte para fazer isso:

```
aws dms create-replication-instance \
    --replication-instance-identifier (the replication instance identifier) \
    --replication-instance-class (the size and capacity of the instance, like
 'dms.t2.medium') \
    --allocated-storage (the number of gigabytes to allocate for the instance
 initially) \
    --engine-version (the DMS engine version that the instance should use) \
    --vpc-security-group-ids (the security group to be used with the instance)
```

Etapa 2. Crie um AWS DMS endpoint para o banco de dados de origem

A próxima etapa é criar um AWS DMS endpoint para seu armazenamento de dados de origem. Você pode usar a AWS DMS CreateEndpointAPI da seguinte AWS CLI forma:

```
aws dms create-endpoint \
    --endpoint-identifier (source endpoint identifier) \
    --endpoint-type source \
    --engine-name (name of source database engine) \
    --username (user name for database login) \
    --password (password for login) \
    --server-name (name of the server) \
    --port (port number) \
    --database-name (database name)
```

Etapa 3. Configurar um bucket do Amazon S3 para o Neptune para uso para dados de preparação

Se você não tiver um bucket do Amazon S3 que possa ser usado para preparar os dados, crie um conforme explicado em Creating a Bucket no Guia de conceitos básicos do Amazon S3 ou How Do I Create an S3 Bucket? no Guia do usuário do console.

Será necessário criar uma política do IAM que conceda permissões `GetObject`, `PutObject`, `DeleteObject` e `ListObject` para o bucket, se você ainda não tiver uma:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListObjectsInBucket",
      "Effect": "Allow",
      "Action": [
```

```
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::(bucket-name)"
      ]
    },
    {
      "Sid": "AllObjectActions",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject",
        "s3:ListObject"
       ],
      "Resource": [
        "arn:aws:s3:::(bucket-name)/*"
      ]
    }
  ]
}
```

Se o cluster de banco de dados do Neptune tiver a autenticação do IAM habilitada, também será necessário incluir a seguinte política:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": "neptune-db:*",
      "Resource": "(the ARN of your Neptune DB cluster resource)"
    }
  ]
}
```

Crie um perfil do IAM como um documento de confiança para associar a política a:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "dms.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    },
    {
      "Sid": "neptune",
      "Effect": "Allow",
      "Principal": {
        "Service": "rds.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Depois de associar a política ao perfil, associe o perfil ao cluster de banco de dados do Neptune. Isso permitirá AWS DMS usar o bucket para preparar os dados que estão sendo carregados.

Etapa 4. Criar um endpoint do Amazon S3 na VPC do Neptune

Agora, crie um endpoint de gateway de VPC para o bucket do Amazon S3 intermediário, na VPC na qual o cluster do Neptune está localizado. Você pode usar o AWS Management Console ou o AWS CLI para fazer isso, conforme descrito em Criação de um endpoint de gateway.

Etapa 5. Crie um endpoint de AWS DMS destino para Neptune

Crie um AWS DMS endpoint para seu cluster de banco de dados Neptune de destino. Você pode usar a AWS DMS CreateEndpointAPI com o `NeptuneSettings` parâmetro como este:

```
aws dms create-endpoint \
    --endpoint-identifier (target endpoint identifier) \
    --endpoint-type target \
    --engine-name neptune \
    --server-name (name of the server) \
    --port (port number) \
    --neptune-settings '{ \
      "ServiceAccessRoleArn": "(ARN of the service access role)", \
      "S3BucketName": "(name of S3 bucket to use for staging files when migrating)", \
      "S3BucketFolder": "(name of the folder to use in that S3 bucket)", \
```

```
      "ErrorRetryDuration": (number of milliseconds to wait between bulk-load retries),
\
      "MaxRetryCount": (the maximum number of times to retry a failing bulk-load job),
\
      "MaxFileSize": (maximum file size, in bytes, of the staging files written to S3),
\
      "isIamAuthEnabled": (set to true if IAM authentication is enabled on the Neptune
cluster) }'
```

O objeto JSON passado para a AWS DMS `CreateEndpoint` API em seu `NeptuneSettings` parâmetro tem os seguintes campos:

- **`ServiceAccessRoleArn`**: (obrigatório) o ARN de um perfil do IAM que permite acesso refinado ao bucket do S3 usado para preparar a migração dos dados para o Neptune. Esse perfil também deverá ter permissões para acessar o cluster de banco de dados do Neptune se a autorização do IAM estiver habilitada nela.
- **`S3BucketName`**: (obrigatório) para a migração de carregamento completo, a instância de replicação converte todos os dados do RDS em arquivos CSV e quad, carrega-os para esse bucket de preparação no S3 e carrega-os em massa para o Neptune.
- **`S3BucketFolder`**: (obrigatório) a pasta a ser usada no bucket de preparação do S3.
- **`ErrorRetryDuration`**: (opcional) o número de milissegundos a aguardar após uma solicitação do Neptune falhar antes de tentar fazer uma nova solicitação. O padrão é 250.
- **`MaxRetryCount`**— (opcional) O número máximo de solicitações de repetição que AWS DMS devem ser feitas após uma falha que possa ser repetida. O padrão é 5.
- **`MaxFileSize`**: (opcional) o tamanho máximo em bytes de cada arquivo de teste salvo no S3 durante a migração. O padrão é 1.048.576 KB (1 GB).
- **`IsIAMAuthEnabled`** (opcional) Defina como `true` defina se a autenticação do IAM estiver habilitada no cluster de banco de dados do Neptune ou `false` se não estiver. O padrão é `false`.

Etapa 6. Testar conexões com os novos endpoints

Você pode testar a conexão com cada um desses novos endpoints usando a AWS DMS TestConnectionAPI da seguinte forma:

```
aws dms test-connection \
    --replication-instance-arn (the ARN of the replication instance) \
```

```
    --endpoint-arn (the ARN of the endpoint you are testing)
```

Etapa 7. Criar uma tarefa de AWS DMS replicação

Depois de concluir as etapas anteriores com êxito, crie uma tarefa de replicação para migrar dados do seu armazenamento de dados de origem para o Neptune, usando a API Task da seguinte forma: AWS DMS CreateReplication

```
aws dms create-replication-task \
    --replication-task-identifier (name for the replication task) \
    --source-endpoint-arn (ARN of the source endpoint) \
    --target-endpoint-arn (ARN of the target endpoint) \
    --replication-instance-arn (ARN of the replication instance) \
    --migration-type full-load \
    --table-mappings (table-mapping JSON object or URI like 'file:///tmp/table-
mappings,json') \
    --task-data (a GraphMappingConfig object or URI like 'file:///tmp/graph-mapping-
config.json')
```

O parâmetro `TaskData` fornece o GraphMappingConfig, que especifica como os dados que estão sendo copiados devem ser armazenados no Neptune.

Etapa 8. Iniciar a tarefa de AWS DMS replicação

Agora, é possível iniciar a tarefa de replicação:

```
aws dms start-replication-task
    --replication-task-arn (ARN of the replication task started in the previous step)
    --start-replication-task-type start-replication
```

Consultar um grafo do Neptune

O Neptune é compatível com as seguintes linguagens de consulta de grafo para acessar um grafo:

- Gremlin, definido pelo Apache TinkerPop para criar e consultar gráficos de propriedades.

  Uma consulta no Gremlin é um percurso composto por etapas distintas, cada uma das quais segue uma borda até um nó.

  Consulte Acessar o grafo do Neptune com o Gremlin para saber como usar o Gremlin no Neptune e Conformidade com os padrões do Gremlin no Amazon Neptunepara encontrar detalhes específicos sobre a implementação do Gremlin no Neptune.
- openCypher é uma linguagem de consulta declarativa para grafos de propriedades originalmente desenvolvida pela Neo4j, que se tornou de código aberto em 2015, e contribuiu para o projeto openCypher sob uma licença de código aberto Apache 2. Sua sintaxe está documentada na especificação do openCypher.
- O SPARQL é uma linguagem declarativa baseada na correspondência de padrões de grafos, para consultar dados do RDF. É apoiado pelo World Wide Web Consortium.

  Consulte Acessar o grafo do Neptune com o SPARQL para saber como usar o SPARQL no Neptune e Conformidade com os padrões SPARQL no Amazon Neptunepara encontrar detalhes específicos sobre a implementação do SPARQL no Neptune.

> **Note**
>
> Tanto o Gremlin quanto o openCypher podem ser usados para consultar quaisquer dados de grafos de propriedades armazenados no Neptune, independentemente de como eles foram carregados.

Tópicos

Colocar consultas na fila no Amazon Neptune

Ao desenvolver e ajustar aplicativos de gráficos, pode ser útil saber as implicações de como as consultas estão sendo colocadas em fila pelo banco de dados. No Amazon Neptune, as consultas são colocadas em fila da seguinte forma:

- O número máximo de consultas que podem ser colocadas em fila por instância, independentemente do tamanho da instância, é 8.192. Quaisquer consultas acima desse número são rejeitadas e falham com `ThrottlingException`.
- O número máximo de consultas que podem ser executadas ao mesmo tempo é determinado pelo número de threads do operador atribuídos, que geralmente é definido como o dobro do número de núcleos de CPU virtuais (vCPUs) disponíveis.
- A latência de consulta inclui o tempo que uma consulta passa na fila, bem como a rota de ida e volta da rede e o tempo que realmente é necessário para ser executada.

Determinar quantas consultas estão em sua fila em um momento específico

A `MainRequestQueuePendingRequests` CloudWatch métrica registra o número de solicitações em espera na fila de entrada em uma granularidade de cinco minutos (consulte). Métricas de Neptune CloudWatch

Para o Gremlin, é possível obter uma contagem atual de consultas na fila usando o valor `acceptedQueryCount` retornado por API de status de consulta do Gremlin. No entanto, observe que o valor `acceptedQueryCount` retornado por API de status de consulta do SPARQL inclui todas as consultas aceitas desde que o servidor foi iniciado, incluindo consultas concluídas.

Como as filas de consultas podem afetar os tempos limite

Conforme observado acima, a latência da consulta inclui o tempo que uma consulta passa na fila e o tempo necessário para ser executada.

Como o tempo limite de uma consulta geralmente é medido com base no momento em que ela entra na fila, uma fila lenta pode fazer com que muitas consultas atinjam o tempo limite assim que forem retiradas da fila. Isso é claramente indesejável, portanto, é bom evitar colocar um grande número de consultas na fila, a menos que elas possam ser executadas rapidamente.

Acessar o grafo do Neptune com o Gremlin

O Amazon Neptune é compatível com TinkerPop Apache 3 e Gremlin. Isso significa que você pode se conectar a uma instância de banco de dados Neptune e usar a linguagem de travessia Gremlin para consultar o gráfico (consulte The Graph na documentação do Apache 3). TinkerPop Para ver as diferenças na implementação do Gremlin no Neptune, consulte Conformidade com os padrões do Gremlin.

Diferentes versões do mecanismo do Neptune são compatíveis com diferentes versões do Gremlin. Confira a página de versão do mecanismo da versão do Neptune que você está executando para determinar qual versão do Gremlin é compatível.

Uma travessia no Gremlin é uma série de etapas encadeadas. Ela começa em um vértice (ou ponto) e percorre o gráfico seguindo os pontos de saída de cada vértice e, em seguida, os pontos de saída desses vértices. Cada etapa é uma operação na travessia. Para obter mais informações, consulte The Traversal na documentação TinkerPop 3.

Há variantes da linguagem Gremlin e suporte para acesso ao Gremlin em várias linguagens de programação. Para obter mais informações, consulte Sobre as variantes da linguagem Gremlin na documentação TinkerPop 3.

Essa documentação descreve como acessar o Neptune com as variantes e linguagens de programação a seguir.

Como discutido em Criptografia em trânsito: estabelecer conexão com o Neptune Using SSL/HTTPS usando SSL/HTTPS, você deve se conectar ao Transport Layer Security/Secure Sockets Layer (TLS/SSL) ao se conectar ao Neptune em todas as regiões da AWS .

Gremlin-Groovy

Os exemplos do console do Gremlin e do REST HTTP desta seção usam a variante Gremlin-Groovy. Para obter mais informações sobre o console do Gremlin e o Amazon Neptune, consulte a seção the section called “Usar o Gremlin” do Quick Start.

Gremlin-Java

A amostra Java foi escrita com a implementação oficial do Java TinkerPop 3 e usa a variante Gremlin-Java.

Gremlin-Python

A amostra do Python foi escrita com a implementação oficial do TinkerPop Python 3 e usa a variante Gremlin-Python.

As seções a seguir explicam como usar o console do Gremlin, REST por meio de HTTPS, e várias linguagens de programação para conectar-se a uma instância de banco de dados do Neptune.

Antes de começar, você deve ter o seguinte:

- Uma instância de banco de dados do Neptune. Para obter informações sobre como criar uma instância de banco de dados do Neptune, consulte Criar um cluster de banco de dados do Neptune.
- A instância do Amazon EC2 na mesma nuvem privada virtual (VPC) que a instância de banco de dados do Neptune.

Para obter mais informações sobre como carregar dados no Neptune, incluindo pré-requisitos, formatos de carregamento e parâmetros de carregamento, consulte Carregar dados no Amazon Neptune.

Tópicos

- Criar upserts eficientes do Gremlin com fold()/coalesce()/unfold()
- Analisar a execução de consulta do Neptune usando o explain do Gremlin
- Usar o Gremlin com o mecanismo de consulta do DFE do Neptune

Configurar o console do Gremlin para conectar-se a uma instância de banco de dados do Neptune

O Gremlin Console permite que você experimente TinkerPop gráficos e consultas em um ambiente REPL (loop). read-eval-print

Instalar o console do Gremlin e conectar-se a ele da maneira usual

Você pode usar o Gremlin Console para conectar-se a um banco de dados de gráficos remoto. A seção a seguir descreve a instalação e a configuração do console do Gremlin para conectar-se remotamente a uma instância de banco de dados do Neptune. Você deve seguir estas instruções em uma instância do Amazon EC2 na mesma nuvem privada virtual (VPC) que a instância de banco de dados do Neptune.

Para obter ajuda na conexão com o Neptune com SSL/TLS (o que é obrigatório), consulte Configuração de SSL/TLS.

> **Note**
>
> Se você tiver a autenticação do IAM habilitada no cluster de banco de dados do Neptune, siga as instruções em Estabelecer conexão com o Neptune usando o console do Gremlin com a assinatura do Signature versão 4 para instalar o console do Gremlin e não as instruções aqui.

Como instalar o console do Gremlin e se conectar ao Neptune

1. Os binários do console do Gremlin exigem o Java 8 ou 11. Essas instruções pressupõem o uso do Java 11. Você pode instalar o Java 11 na instância do EC2 da seguinte forma:

   - Se você estiver usando o Amazon Linux 2 (AL2):

     ```
     sudo amazon-linux-extras install java-openjdk11
     ```

- Se você estiver usando o Amazon Linux 2023 (AL2023):

```
sudo yum install java-11-amazon-corretto-devel
```

- Para outras distribuições, use qualquer uma das seguintes opções que seja apropriada:

```
sudo yum install java-11-openjdk-devel
```

ou:

```
sudo apt-get install openjdk-11-jdk
```

2. Insira o seguinte para definir o Java 11 como o runtime padrão na instância do EC2.

```
sudo /usr/sbin/alternatives --config java
```

Quando solicitado, insira o número do Java 11.

3. Baixe a versão apropriada do console do Gremlin pelo site da Apache. É possível conferir a página de versão do mecanismo da versão do mecanismo do Neptune que você está executando no momento para determinar qual versão do Gremlin é compatível. Por exemplo, para a versão 3.6.5, você pode baixar o console do Gremlin pelo site do Apache Tinkerpop3 na instância do EC2 da seguinte forma:

```
wget https://archive.apache.org/dist/tinkerpop/3.6.5/apache-tinkerpop-gremlin-
console-3.6.5-bin.zip
```

4. Descompacte o arquivo zip do Gremlin Console.

```
unzip apache-tinkerpop-gremlin-console-3.6.5-bin.zip
```

5. Altere os diretórios para o diretório descompactado.

```
cd apache-tinkerpop-gremlin-console-3.6.5
```

6. No subdiretório `conf` do diretório extraído, crie um arquivo chamado `neptune-remote.yaml` com o seguinte texto. Substitua *`your-neptune-endpoint`* pelo nome de host ou endereço IP da instância de banco de dados do Neptune. Os colchetes (`[ ]`) são necessários.

> **Note**
>
> Para obter informações sobre como localizar o nome do host da instância de banco de dados do Neptune, consulte a seção Conectar-se a endpoints do Amazon Neptune.

```
hosts: [your-neptune-endpoint]
port: 8182
connectionPool: { enableSsl: true }
serializer: { className:
 org.apache.tinkerpop.gremlin.util.ser.GraphBinaryMessageSerializerV1,
              config: { serializeResultToString: true }}
```

> **Note**
>
> Os serializadores foram movidos do `gremlin-driver` módulo para o novo `gremlin-util` módulo na versão 3.7.0. O pacote mudou de org.apache.tinkerpop.gremlin.driver.ser para org.apache.tinkerpop.gremlin.util.ser.

7. Em um terminal, navegue até o diretório do Gremlin Console (`apache-tinkerpop-gremlin-console-3.6.5`) e, em seguida, insira o comando a seguir para executar o Gremlin Console.

```
bin/gremlin.sh
```

A seguinte saída deverá ser mostrada:

```
         \,,,/
         (o o)
-----oOOo-(3)-oOOo-----
plugin activated: tinkerpop.server
plugin activated: tinkerpop.utilities
plugin activated: tinkerpop.tinkergraph
gremlin>
```

Agora você está no prompt do `gremlin>`. Você inserirá as etapas restantes nesse prompt.

8. No prompt `gremlin>`, insira o seguinte para conectar-se a instâncias de banco de dados do Neptune.

```
:remote connect tinkerpop.server conf/neptune-remote.yaml
```

9. No prompt `gremlin>`, insira o seguinte para alternar para modo remoto. Isso envia todas as consultas do Gremlin para a conexão remota.

```
:remote console
```

10. Insira o seguinte para enviar uma consulta para o Gremlin Graph.

```
g.V().limit(1)
```

11. Ao terminar, insira o seguinte para sair do Gremlin Console.

```
:exit
```

> **Note**
>
> Use um ponto e vírgula (`;`) ou um caractere de nova linha (`\n`) para separar cada instrução. Cada travessia anterior à travessia final deve terminar em `next()` a ser executada. Somente os dados da travessia final são retornados.

Para obter mais informações sobre a implementação do Gremlin no Neptune, consulte the section called “Conformidade com os padrões do Gremlin”.

Uma forma alternativa de se conectar ao console do Gremlin

Desvantagens da abordagem de conexão normal

A forma mais comum de se conectar ao console do Gremlin é a explicada acima, usando comandos como este no prompt `gremlin>`:

```
gremlin> :remote connect tinkerpop.server conf/(file name).yaml
gremlin> :remote console
```

Isso funciona bem e permite que você envie consultas ao Neptune. No entanto, retira o mecanismo de script Groovy do circuito, então, o Neptune trata todas as consultas como Gremlin puro. Isso significa que os seguintes formatos de consulta falham:

```
gremlin> 1 + 1
gremlin> x = g.V().count()
```

O mais próximo que você pode chegar de usar uma variável quando conectado dessa maneira é usar a variável `result` mantida pelo console e enviar a consulta usando `:>`, desta forma:

```
gremlin> :remote console
==>All scripts will now be evaluated locally - type ':remote console' to return
 to remote mode for Gremlin Server - [krl-1-cluster.cluster-ro-cm9t6tfwbtsr.us-
east-1.neptune.amazonaws.com/172.31.19.217:8182]
gremlin> :> g.V().count()
==>4249

gremlin> println(result)
[result{object=4249 class=java.lang.Long}]

gremlin> println(result['object'])
[4249]
```

Uma forma diferente de se conectar

Também é possível conectar-se ao console do Gremlin de outra maneira talvez mais agradável, desta forma:

```
gremlin> g = traversal().withRemote('conf/neptune.properties')
```

Aqui `neptune.properties` assume o seguinte formato:

```
gremlin.remote.remoteConnectionClass=org.apache.tinkerpop.gremlin.driver.remote.DriverRemoteCon
gremlin.remote.driver.clusterFile=conf/my-cluster.yaml
gremlin.remote.driver.sourceName=g
```

O arquivo `my-cluster.yaml` deve ser semelhante a este:

```
hosts: [my-cluster-abcdefghijk.us-east-1.neptune.amazonaws.com]
port: 8182
serializer: { className:
 org.apache.tinkerpop.gremlin.util.ser.GraphBinaryMessageSerializerV1,
              config: { serializeResultToString: false } }
```

```
connectionPool: { enableSsl: true }
```

> **Note**
>
> Os serializadores foram movidos do `gremlin-driver` módulo para o novo `gremlin-util` módulo na versão 3.7.0. O pacote mudou de org.apache.tinkerpop.gremlin.driver.ser para org.apache.tinkerpop.gremlin.util.ser.

Configurar a conexão do console do Gremlin dessa forma permite fazer os seguintes tipos de consulta com êxito:

```
gremlin> 1+1
==>2

gremlin> x=g.V().count().next()
==>4249

gremlin> println("The answer was ${x}")
The answer was 4249
```

Você pode evitar a exibição do resultado da seguinte maneira:

```
gremlin> x=g.V().count().next();[]
gremlin> println(x)
4249
```

Todas as formas comuns de consulta (sem a etapa terminal) continuam funcionando. Por exemplo: .

```
gremlin> g.V().count()
==>4249
```

Você pode até usar a etapa `g.io().read()` para carregar um arquivo com esse tipo de conexão.

Usar o endpoint REST HTTP para conectar-se a uma instância de banco de dados do Neptune

O Amazon Neptune fornece um endpoint HTTP para consultas do Gremlin. A interface REST é compatível com qualquer versão do Gremlin que o cluster de banco de dados esteja usando

(consulte a página de versão do mecanismo da versão do mecanismo do Neptune que você está executando para determinar qual versão do Gremlin é compatível).

> **Note**
>
> Conforme abordado em Criptografia em trânsito: estabelecer conexão com o Neptune Using SSL/HTTPS usando SSL/HTTPS, o Neptune agora exige que você se conecte usando HTTPS em vez de HTTP.

As instruções a seguir explicam como conectar-se ao endpoint do Gremlin usando o comando `curl` e HTTPS. Você deve seguir estas instruções em uma instância do Amazon EC2 na mesma nuvem privada virtual (VPC) que a instância de banco de dados do Neptune.

O endpoint HTTP para consultas do Gremlin para uma instância de banco de dados do Neptune é `https://`*`your-neptune-endpoint`*`:`*`port`*`/gremlin`.

> **Note**
>
> Para obter informações sobre como localizar o nome do host da instância de banco de dados do Neptune, consulte Conectar-se a endpoints do Amazon Neptune.

Como conectar-se ao Neptune usando o endpoint REST HTTP

O exemplo a seguir usa o curl para enviar uma consulta do Gremlin por meio de HTTP POST. A consulta é enviada no formato JSON no corpo da postagem como a propriedade `gremlin`.

```
curl -X POST -d '{"gremlin":"g.V().limit(1)"}' https://your-neptune-endpoint:port/
gremlin
```

Este exemplo exibe o primeiro vértice do grafo usando o percurso `g.V().limit(1)`. É possível consultar outro elemento substituindo-o por outro percurso do Gremlin.

> **Important**
>
> Por padrão, o endpoint REST exibe todos os resultados em um único conjunto de resultados JSON. Se esse conjunto de resultados for grande demais, uma exceção `OutOfMemoryError` poderá ocorrer na instância de banco de dados do Neptune.

É possível evitar isso habilitando respostas fragmentadas (resultados gerados em uma série de respostas separadas). Consulte Usar cabeçalhos finais HTTP opcionais para habilitar respostas do Gremlin em várias partes.

Embora as solicitações HTTP POST sejam recomendadas para enviar consultas do Gremlin, também é possível usar solicitações HTTP GET:

```
curl -G "https://your-neptune-endpoint:port?gremlin=g.V().count()"
```

> **Note**
>
> O Neptune não é compatível com a propriedade `bindings`.

Usar cabeçalhos finais HTTP opcionais para habilitar respostas do Gremlin em várias partes

Por padrão, a resposta HTTP às consultas do Gremlin é gerada em um único conjunto de resultados JSON. No caso de um conjunto de resultados muito grande, isso pode causar uma exceção `OutOfMemoryError` na instância de banco de dados.

No entanto, é possível habilitar respostas fragmentadas (respostas geradas em várias partes separadas). Isso é feito incluindo um cabeçalho de trailers (`te: trailers`) com codificação de transferência (TE) na solicitação. Consulte a página MDN sobre cabeçalhos de solicitação TE para obter mais informações sobre cabeçalhos TE.

Quando uma resposta é gerada em várias partes, pode ser difícil diagnosticar um problema ocorrido após o recebimento da primeira parte, pois a primeira parte chega com um código de status HTTP de `200` (OK). Uma falha subsequente geralmente gera um corpo de mensagem que contém uma resposta corrompida, ao final da qual o Neptune acrescenta uma mensagem de erro.

Para facilitar a detecção e o diagnóstico desse tipo de falha, o Neptune também inclui dois novos campos de cabeçalho nos cabeçalhos finais de cada bloco de resposta:

- `X-Neptune-Status`: contém o código de resposta seguido por um nome curto. Por exemplo, em caso de êxito, o cabeçalho final seria: `X-Neptune-Status: 200 OK`. Em caso de falha, o código de resposta seria um dos códigos de erro do mecanismo do Neptune, como `X-Neptune-Status: 500 TimeLimitExceededException`.

- `X-Neptune-Detail`: fica em branco para solicitações bem-sucedidas. No caso de erros, ele contém a mensagem de erro JSON. Como somente caracteres ASCII são permitidos nos valores do cabeçalho HTTP, a string JSON é codificada em URL.

> **Note**
>
> No momento, o Neptune não é compatível com a compactação `gzip` de respostas fragmentadas. Se o cliente solicitar codificação fragmentada e compactação ao mesmo tempo, o Neptune vai ignorar a compactação.

Clientes do Gremlin baseados em Java para usar com o Amazon Neptune

Você pode usar qualquer um dos dois clientes Gremlin de código aberto baseados em Java com o Amazon Neptune: o cliente Apache TinkerPop Java Gremlin ou o cliente Gremlin para o Amazon Neptune.

Cliente Apache TinkerPop Java Gremlin

Se você puder, sempre use a versão mais recente do cliente Apache TinkerPop Java Gremlin compatível com a versão do seu mecanismo. As versões mais recentes contêm várias correções de erros que podem melhorar a estabilidade, o desempenho e a usabilidade do cliente.

A tabela abaixo lista as versões mais antigas e mais recentes do TinkerPop cliente suportadas pelas diferentes versões do Neptune Engine:

Versão do mecanismo do Neptune	TinkerPop Versão mínima	TinkerPop Versão máxima
`1.3.2.0`	`3.6.2`	`3.7.1`
`1.3.1.0`	`3.6.2`	`3.6.5`
`1.3.0.0`	`3.6.2`	`3.6.4`
`1.2.1.1`	`3.6.2`	`3.6.2`
`1.2.1.0`	`3.6.2`	`3.6.2`

Versão do mecanismo do Neptune	TinkerPop Versão mínima	TinkerPop Versão máxima
1.2.0.2	3.5.2	3.5.6
1.2.0.1	3.5.2	3.5.6
1.2.0.0	3.5.2	3.5.6
1.1.1.0	3.5.2	3.5.6
1.1.0.0	3.4.0	3.4.13
1.0.5.1 e posterior	(obsoleto)	(obsoleto)

TinkerPop os clientes geralmente são compatíveis com versões anteriores em uma série (3.3.x, por exemplo, ou3.4.x). Há casos excepcionais em que a compatibilidade com versões anteriores precisa ser quebrada, então é melhor verificar a recomendação de TinkerPop atualização antes de atualizar para uma nova versão do cliente.

Talvez o cliente não consiga usar novas etapas ou novos recursos introduzidos em versões posteriores às compatíveis com o servidor, mas é possível esperar que as consultas e os atributos existentes funcionem, a menos que a recomendação de atualização indique uma alteração significativa.

> **Note**
>
> A partir da versão 1.1.1.0 do motor Neptune, não use uma versão inferior a. TinkerPop 3.5.2
> Os usuários do Python devem evitar usar a TinkerPop versão 3.4.9 devido a uma configuração de tempo limite padrão que requer configuração direta (consulte TINKERPOP-2505).

Cliente Java do Gremlin para Amazon Neptune

O cliente Gremlin para o Amazon Neptune é um cliente Gremlin de código aberto baseado em Java que atua como um substituto imediato para o cliente Java padrão. TinkerPop

O cliente do Gremlin no Neptune é otimizado para clusters do Neptune. Ele permite que você gerencie a distribuição de tráfego em várias instâncias em um cluster e se adapte às alterações na topologia do cluster quando você adiciona ou remove uma réplica. É possível até mesmo configurar o cliente para distribuir solicitações em um subconjunto de instâncias no cluster, com base no perfil, no tipo de instância, na zona de disponibilidade (AZ) ou em tags associadas às instâncias.

A versão mais recente do cliente Java do Gremlin do Neptune está disponível no Maven Central.

Para obter mais informações sobre o cliente Java do Gremlin do Neptune, consulte esta postagem no blog. Para exemplos de código e demonstrações, confira o GitHub projeto do cliente.

Usar o cliente do Java para conectar-se a uma instância de banco de dados do Neptune

A seção a seguir mostra a execução de uma amostra completa de Java que se conecta a uma instância de banco de dados Neptune e executa uma travessia do Gremlin usando o cliente Apache Gremlin. TinkerPop

Estas instruções devem ser seguidas em uma instância do Amazon EC2 na mesma nuvem privada virtual (VPC) que a instância de banco de dados do Neptune.

Como conectar-se ao Neptune usando o Java

1. Instale o Apache Maven na instância do EC2. Primeiro, insira o seguinte para adicionar um repositório com um pacote do Maven:

   ```
   sudo wget https://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-apache-maven.repo
   ```

   Insira o seguinte para definir o número da versão para os pacotes:

   ```
   sudo sed -i s/\$releasever/6/g /etc/yum.repos.d/epel-apache-maven.repo
   ```

   Em seguida, use yum para instalar Maven:

   ```
   sudo yum install -y apache-maven
   ```

2. Instale o Java. As bibliotecas do Gremlin precisam do Java 8 ou 11. Você pode instalar o Java 11 da seguinte maneira:

- Se você estiver usando o Amazon Linux 2 (AL2):

```
sudo amazon-linux-extras install java-openjdk11
```

- Se você estiver usando o Amazon Linux 2023 (AL2023):

```
sudo yum install java-11-amazon-corretto-devel
```

- Para outras distribuições, use qualquer uma das seguintes opções que seja apropriada:

```
sudo yum install java-11-openjdk-devel
```

ou:

```
sudo apt-get install openjdk-11-jdk
```

3. Defina o Java 11 como o runtime padrão na instância do EC2: digite o seguinte para definir o Java 8 como o runtime padrão na instância do EC2:

```
sudo /usr/sbin/alternatives --config java
```

Quando solicitado, insira o número do Java 11.

4. Crie um diretório denominado **`gremlinjava`**:

```
mkdir gremlinjava
cd gremlinjava
```

5. No diretório `gremlinjava`, crie um arquivo `pom.xml` e abra-o em um editor de texto:

```
nano pom.xml
```

6. Copie o seguinte no arquivo `pom.xml` e salve-o:

```
<project xmlns="https://maven.apache.org/POM/4.0.0"
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="https://maven.apache.org/POM/4.0.0 https://
maven.apache.org/maven-v4_0_0.xsd">
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
```

```
  </properties>
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.amazonaws</groupId>
  <artifactId>GremlinExample</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>GremlinExample</name>
  <url>https://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>org.apache.tinkerpop</groupId>
      <artifactId>gremlin-driver</artifactId>
      <version>3.6.5</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.apache.tinkerpop/gremlin-groovy
      (Not needed for TinkerPop version 3.5.2 and up)
    <dependency>
      <groupId>org.apache.tinkerpop</groupId>
      <artifactId>gremlin-groovy</artifactId>
      <version>3.6.5</version>
    </dependency> -->
    <dependency>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-jdk14</artifactId>
      <version>1.7.25</version>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>2.5.1</version>
        <configuration>
          <source>11</source>
          <target>11</target>
        </configuration>
      </plugin>
        <plugin>
          <groupId>org.codehaus.mojo</groupId>
          <artifactId>exec-maven-plugin</artifactId>
          <version>1.3</version>
          <configuration>
            <executable>java</executable>
```

```
            <arguments>
              <argument>-classpath</argument>
              <classpath/>
              <argument>com.amazonaws.App</argument>
            </arguments>
            <mainClass>com.amazonaws.App</mainClass>
            <complianceLevel>1.11</complianceLevel>
            <killAfter>-1</killAfter>
          </configuration>
        </plugin>
    </plugins>
  </build>
</project>
```

> **Note**
>
> Se estiver modificando um projeto existente do Maven, a dependência necessária será destacada no código anterior.

7. Crie subdiretórios para o código-fonte de exemplo (`src/main/java/com/amazonaws/`) digitando o seguinte na linha de comando:

```
mkdir -p src/main/java/com/amazonaws/
```

8. No diretório `src/main/java/com/amazonaws/`, crie um arquivo denominado `App.java` e abra-o em um editor de texto.

```
nano src/main/java/com/amazonaws/App.java
```

9. Copie o seguinte no arquivo `App.java`. Substitua *`your-neptune-endpoint`* pelo endereço da instância de banco de dados do Neptune. Não inclua o prefixo `https://` no método `addContactPoint`.

> **Note**
>
> Para obter informações sobre como localizar o nome do host da instância de banco de dados do Neptune, consulte Conectar-se a endpoints do Amazon Neptune.

```
package com.amazonaws;
```

```
import org.apache.tinkerpop.gremlin.driver.Cluster;
import org.apache.tinkerpop.gremlin.driver.Client;
import
 org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversalSource;
import org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversal;
import static
 org.apache.tinkerpop.gremlin.process.traversal.AnonymousTraversalSource.traversal;
import org.apache.tinkerpop.gremlin.driver.remote.DriverRemoteConnection;
import org.apache.tinkerpop.gremlin.structure.T;

public class App
{
  public static void main( String[] args )
  {
    Cluster.Builder builder = Cluster.build();
    builder.addContactPoint("your-neptune-endpoint");
    builder.port(8182);
    builder.enableSsl(true);

    Cluster cluster = builder.create();

    GraphTraversalSource g =
 traversal().withRemote(DriverRemoteConnection.using(cluster));

    // Add a vertex.
    // Note that a Gremlin terminal step, e.g. iterate(), is required to make a
 request to the remote server.
    // The full list of Gremlin terminal steps is at https://tinkerpop.apache.org/
docs/current/reference/#terminal-steps
    g.addV("Person").property("Name", "Justin").iterate();

    // Add a vertex with a user-supplied ID.
    g.addV("Custom Label").property(T.id, "CustomId1").property("name", "Custom id
 vertex 1").iterate();
    g.addV("Custom Label").property(T.id, "CustomId2").property("name", "Custom id
 vertex 2").iterate();

    g.addE("Edge Label").from(__.V("CustomId1")).to(__.V("CustomId2")).iterate();

    // This gets the vertices, only.
    GraphTraversal t = g.V().limit(3).elementMap();

    t.forEachRemaining(
      e ->  System.out.println(t.toList())
```

```
    );

    cluster.close();
  }
}
```

Para obter ajuda na conexão com o Neptune com SSL/TLS (o que é obrigatório), consulte Configuração de SSL/TLS.

10. Compile e execute o exemplo usando o seguinte comando do Maven:

```
mvn compile exec:exec
```

O exemplo anterior retorna um mapa da chave e dos valores de cada propriedade dos dois primeiros vértices no gráfico usando a travessia `g.V().limit(3).elementMap()`. Para consultar outro elemento, substitua-a por outra travessia do Gremlin com um dos métodos de término adequado.

> **Note**
>
> A parte final da consulta do Gremlin, `.toList()`, é necessária para enviar a travessia ao servidor para avaliação. Se você não incluir esse método ou outro método equivalente, a consulta não será enviada à instância de banco de dados do Neptune.
> Você também deve associar uma terminação adequada ao adicionar um vértice ou uma borda, por exemplo, ao usar a etapa `addV( )`.

Os seguintes métodos enviam a consulta à instância de banco de dados do Neptune:

- `toList()`
- `toSet()`
- `next()`
- `nextTraverser()`
- `iterate()`

Configuração de SSL/TLS para o cliente Java do Gremlin

O Neptune exige que o SSL/TLS esteja habilitado por padrão. Normalmente, se o driver Java estiver configurado com `enableSsl(true)`, ele poderá se conectar ao Neptune sem precisar

configurar um `trustStore()` ou um `keyStore()` com uma cópia local de um certificado. As versões anteriores do TinkerPop incentivavam o uso do `keyCertChainFile()` para configurar um `.pem` arquivo armazenado localmente, mas isso se tornou obsoleto e não está mais disponível após a versão 3.5.x. Se você estava usando essa configuração com um certificado público, usando `SFSRootCAG2.pem`, agora você pode remover a cópia local.

No entanto, se a instância com a qual você está se conectando não tiver conexão com a Internet para verificar um certificado público, ou se o certificado utilizado não for público, siga estas etapas para configurar uma cópia do certificado local:

Configurar uma cópia de certificado local para habilitar SSL/TLS

1. Baixe e instale o keytool da Oracle. Isso facilitará muito a configuração do armazenamento de chaves local.
2. Baixe um certificado CA `SFSRootCAG2.pem` (o SDK Java do Gremlin exige um certificado para verificar o certificado remoto):

   ```
   wget https://www.amazontrust.com/repository/SFSRootCAG2.pem
   ```

3. Crie um armazenamento de chaves no formato JKS ou PKCS12. Este exemplo usa JKS. Responda às perguntas que se seguem no prompt. A senha que você criar aqui será necessária posteriormente:

   ```
   keytool -genkey -alias (host name) -keyalg RSA -keystore server.jks
   ```

4. Importe o arquivo `SFSRootCAG2.pem` que você baixou para o repositório de chaves recém-criado:

   ```
   keytool -import -keystore server.jks -file .pem
   ```

5. Configure o objeto `Cluster` programaticamente:

   ```
   Cluster cluster = Cluster.build("(your neptune endpoint)")
                            .port(8182)
                            .enableSSL(true)
                            .keyStore('server.jks')
                            .keyStorePassword("(the password from step 2)")
                            .create();
   ```

Se desejar, você poderá fazer o mesmo em um arquivo de configuração, como faria com o console do Gremlin:

```
hosts: [(your neptune endpoint)]
port: 8182
connectionPool: { enableSsl: true, keyStore: server.jks, keyStorePassword: (the
 password from step 2) }
serializer: { className:
 org.apache.tinkerpop.gremlin.driver.ser.GraphBinaryMessageSerializerV1, config:
 { serializeResultToString: true }}
```

Exemplo em Java de conexão com uma instância de banco de dados do Neptune com lógica de reconexão

O exemplo de Java a seguir demonstra como se conectar ao cliente do Gremlin com a lógica de reconexão para se recuperar de uma desconexão inesperada.

As seguintes dependências se aplicam:

```
<dependency>
    <groupId>org.apache.tinkerpop</groupId>
    <artifactId>gremlin-driver</artifactId>
    <version>${gremlin.version}</version>
</dependency>

<dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>amazon-neptune-sigv4-signer</artifactId>
    <version>${sig4.signer.version}</version>
</dependency>

<dependency>
    <groupId>com.evanlennick</groupId>
    <artifactId>retry4j</artifactId>
    <version>0.15.0</version>
</dependency>
```

Aqui está o exemplo de código:

```
public static void main(String args[]) {
```

```
 boolean useIam = true;

 // Create Gremlin cluster and traversal source
 Cluster.Builder builder = Cluster.build()
        .addContactPoint(System.getenv("neptuneEndpoint"))
        .port(Integer.parseInt(System.getenv("neptunePort")))
        .enableSsl(true)
        .minConnectionPoolSize(1)
        .maxConnectionPoolSize(1)
        .serializer(Serializers.GRAPHBINARY_V1D0)
        .reconnectInterval(2000);

 if (useIam) {
     builder.handshakeInterceptor( r -> {
        try {
           NeptuneNettyHttpSigV4Signer sigV4Signer =
                       new NeptuneNettyHttpSigV4Signer("(your region)", new
DefaultAWSCredentialsProviderChain());
           sigV4Signer.signRequest(r);
        } catch (NeptuneSigV4SignerException e) {
           throw new RuntimeException("Exception occurred while signing the request",
e);
        }
        return r;
     });
  }

 Cluster cluster = builder.create();

 GraphTraversalSource g = AnonymousTraversalSource
     .traversal()
     .withRemote(DriverRemoteConnection.using(cluster));

 // Configure retries
 RetryConfig retryConfig = new RetryConfigBuilder()
     .retryOnCustomExceptionLogic(getRetryLogic())
     .withDelayBetweenTries(1000, ChronoUnit.MILLIS)
     .withMaxNumberOfTries(5)
     .withFixedBackoff()
     .build();

 @SuppressWarnings("unchecked")
 CallExecutor<Object> retryExecutor = new CallExecutorBuilder<Object>()
     .config(retryConfig)
```

```
      .build();

  // Do lots of queries
  for (int i = 0; i < 100; i++){
    String id = String.valueOf(i);

    @SuppressWarnings("unchecked")
    Callable<Object> query = () -> g.V(id)
        .fold()
        .coalesce(
            unfold(),
            addV("Person").property(T.id, id))
        .id().next();

    // Retry query
    // If there are connection failures, the Java Gremlin client will automatically
    // attempt to reconnect in the background, so all we have to do is wait and retry.
    Status<Object> status = retryExecutor.execute(query);

    System.out.println(status.getResult().toString());
  }

  cluster.close();
}

private static Function<Exception, Boolean> getRetryLogic() {

  return e -> {

    Class<? extends Exception> exceptionClass = e.getClass();

    StringWriter stringWriter = new StringWriter();
    String message = stringWriter.toString();


    if (RemoteConnectionException.class.isAssignableFrom(exceptionClass)){
      System.out.println("Retrying because RemoteConnectionException");
      return true;
    }

    // Check for connection issues
    if (message.contains("Timed out while waiting for an available host") ||
        message.contains("Timed-out") && message.contains("waiting for connection on
 Host") ||
```

```
        message.contains("Connection to server is no longer active") ||
        message.contains("Connection reset by peer") ||
        message.contains("SSLEngine closed already") ||
        message.contains("Pool is shutdown") ||
        message.contains("ExtendedClosedChannelException") ||
        message.contains("Broken pipe") ||
        message.contains(System.getenv("neptuneEndpoint")))
    {
      System.out.println("Retrying because connection issue");
      return true;
    };

    // Concurrent writes can sometimes trigger a ConcurrentModificationException.
    // In these circumstances you may want to backoff and retry.
    if (message.contains("ConcurrentModificationException")) {
      System.out.println("Retrying because ConcurrentModificationException");
      return true;
    }

    // If the primary fails over to a new instance, existing connections to the old
 primary will
    // throw a ReadOnlyViolationException. You may want to back and retry.
    if (message.contains("ReadOnlyViolationException")) {
      System.out.println("Retrying because ReadOnlyViolationException");
      return true;
    }

    System.out.println("Not a retriable error");
    return false;
  };
}
```

Usar o Python para conectar-se a uma instância de banco de dados do Neptune

Se você puder, sempre use a versão mais recente do cliente Apache TinkerPop Python Gremlin, gremlinpython, compatível com a versão do seu mecanismo. As versões mais recentes contêm várias correções de erros que melhoram a estabilidade, o desempenho e a usabilidade do cliente. A `gremlinpython` versão a ser usada normalmente se alinhará às TinkerPop versões descritas na tabela do cliente Java Gremlin.

> **Note**
>
> As versões `gremlinpython` 3.5.x são compatíveis com as versões TinkerPop 3.4.x, desde que você use apenas os recursos 3.4.x nas consultas do Gremlin que você escreve.

A seção a seguir descreve a execução de um exemplo do Python que se conecta a uma instância de banco de dados do Amazon Neptune e executa um percurso do Gremlin.

Você deve seguir estas instruções em uma instância do Amazon EC2 na mesma nuvem privada virtual (VPC) que a instância de banco de dados do Neptune.

Antes de começar, faça o seguinte:

- Baixe e instale o Python 3.6 ou posterior do site Python.org.
- Verifique se você tem o pip instalado. Se não tiver o pip ou não tiver certeza, consulte Do I need to install pip? (Preciso instalar pip?) na pip documentação.
- Se sua instalação do Python não o tiver, faça download de `futures` da seguinte forma: `pip install futures`

Como conectar-se ao Neptune usando o Python

1. Insira o seguinte para instalar o pacote `gremlinpython`:

   ```
   pip install --user gremlinpython
   ```

2. Crie um arquivo denominado `gremlinexample.py` e abra-o em um editor de texto.
3. Copie o seguinte no arquivo `gremlinexample.py`. Substitua *`your-neptune-endpoint`* pelo endereço da instância de banco de dados do Neptune.

   Para obter informações sobre como localizar o endereço da instância de banco de dados do Neptune, consulte a seção Conectar-se a endpoints do Amazon Neptune.

   ```
   from __future__  import print_function  # Python 2/3 compatibility

   from gremlin_python import statics
   from gremlin_python.structure.graph import Graph
   from gremlin_python.process.graph_traversal import __
   ```

```
from gremlin_python.process.strategies import *
from gremlin_python.driver.driver_remote_connection import DriverRemoteConnection

graph = Graph()

remoteConn = DriverRemoteConnection('wss://your-neptune-endpoint:8182/gremlin','g')
g = graph.traversal().withRemote(remoteConn)

print(g.V().limit(2).toList())
remoteConn.close()
```

4. Insira o seguinte comando para executar o exemplo:

```
python gremlinexample.py
```

A consulta do Gremlin no final deste exemplo retorna os vértices (`g.V().limit(2)`) em uma lista. Em seguida, essa lista é impressa com a função `print` padrão do Python.

> **Note**
>
> A parte final da consulta do Gremlin, `toList()`, é necessária para enviar a travessia ao servidor para avaliação. Se você não incluir esse método ou outro método equivalente, a consulta não será enviada à instância de banco de dados do Neptune.

Os seguintes métodos enviam a consulta à instância de banco de dados do Neptune:

- `toList()`
- `toSet()`
- `next()`
- `nextTraverser()`
- `iterate()`

O exemplo anterior retorna os primeiros dois vértices do gráfico usando a travessia `g.V().limit(2).toList()`. Para consultar outro elemento, substitua-a por outra travessia do Gremlin com um dos métodos de término adequado.

Usar .NET para conectar-se a uma instância de banco de dados do Neptune

Se você puder, sempre use a versão mais recente do cliente Apache TinkerPop .NET Gremlin, Gremlin.Net, compatível com a versão do seu mecanismo. As versões mais recentes contêm várias correções de erros que melhoram a estabilidade, o desempenho e a usabilidade do cliente. A `Gremlin.Net` versão a ser usada normalmente se alinhará às TinkerPop versões descritas na tabela do cliente Java Gremlin.

A seção a seguir contém um exemplo de código escrito em C# que se conecta a uma instância de banco de dados do Neptune e executa um percurso do Gremlin.

As conexões com o Amazon Neptune devem ser de uma instância do Amazon EC2 na mesma nuvem privada virtual (VPC) que a instância de banco de dados do Neptune. Este exemplo de código foi testado em uma instância do Amazon EC2 executando o Ubuntu.

Antes de começar, faça o seguinte:

- Instale o .NET na instância do Amazon EC2. Para obter instruções para instalação do .NET em vários sistemas operacionais, inclusive o Windows, o Linux e o macOS, consulte Conceitos básicos do .NET.
- Instale o Gremlin.NET executando `dotnet add package gremlin.net` para seu pacote. Para obter mais informações, consulte Gremlin.net na documentação. TinkerPop

Como conectar-se ao Neptune usando o Gremlin.NET

1. Crie um novo projeto .NET.

```
dotnet new console -o gremlinExample
```

2. Altere os diretórios para o diretório do novo projeto.

```
cd gremlinExample
```

3. Copie o seguinte no arquivo `Program.cs`. Substitua *`your-neptune-endpoint`* pelo endereço da instância de banco de dados do Neptune.

Para obter informações sobre como localizar o endereço da instância de banco de dados do Neptune, consulte a seção Conectar-se a endpoints do Amazon Neptune.

```
using System;
using System.Threading.Tasks;
using System.Collections.Generic;
using Gremlin.Net;
using Gremlin.Net.Driver;
using Gremlin.Net.Driver.Remote;
using Gremlin.Net.Structure;
using static Gremlin.Net.Process.Traversal.AnonymousTraversalSource;
namespace gremlinExample
{
  class Program
  {
    static void Main(string[] args)
    {
      try
      {
        var endpoint = "your-neptune-endpoint";
        // This uses the default Neptune and Gremlin port, 8182
        var gremlinServer = new GremlinServer(endpoint, 8182, enableSsl: true );
        var gremlinClient = new GremlinClient(gremlinServer);
        var remoteConnection = new DriverRemoteConnection(gremlinClient, "g");
        var g = Traversal().WithRemote(remoteConnection);
        g.AddV("Person").Property("Name", "Justin").Iterate();
        g.AddV("Custom Label").Property("name", "Custom id vertex 1").Iterate();
        g.AddV("Custom Label").Property("name", "Custom id vertex 2").Iterate();
        var output = g.V().Limit<Vertex>(3).ToList();
        foreach(var item in output) {
            Console.WriteLine(item);
        }
      }
      catch (Exception e)
      {
          Console.WriteLine("{0}", e);
      }
    }
  }
}
```

4. Insira o seguinte comando para executar o exemplo:

```
dotnet run
```

A consulta do Gremlin no final deste exemplo retorna a contagem de um único vértice para fins de teste. Em seguida, ela é impressa no console.

> **Note**
>
> A parte final da consulta do Gremlin, `Next()`, é necessária para enviar a travessia ao servidor para avaliação. Se você não incluir esse método ou outro método equivalente, a consulta não será enviada à instância de banco de dados do Neptune.

Os seguintes métodos enviam a consulta à instância de banco de dados do Neptune:

- `ToList()`
- `ToSet()`
- `Next()`
- `NextTraverser()`
- `Iterate()`

Use `Next()` se precisar que os resultados da consulta sejam serializados e gerados, ou `Iterate()` se não precisar.

O exemplo anterior gera uma lista usando o percurso `g.V().Limit(3).ToList()`. Para consultar outro elemento, substitua-a por outra travessia do Gremlin com um dos métodos de término adequado.

Usar o Node.js para conectar-se a uma instância de banco de dados do Neptune

Se você puder, sempre use a versão mais recente do cliente Apache TinkerPop JavaScript Gremlin, gremlin, compatível com a versão do seu mecanismo. As versões mais recentes contêm várias correções de erros que melhoram a estabilidade, o desempenho e a usabilidade do cliente. A versão `gremlin` a ser usada normalmente se alinha às TinkerPop versões descritas na tabela do cliente Java Gremlin.

A seção a seguir descreve a execução de um exemplo do Node.js que se conecta a uma instância de banco de dados do Amazon Neptune e executa um percurso do Gremlin.

Você deve seguir estas instruções em uma instância do Amazon EC2 na mesma nuvem privada virtual (VPC) que a instância de banco de dados do Neptune.

Antes de começar, faça o seguinte:

- Verifique se a versão Node.js 8.11 ou superior está instalada. Se não estiver, faça download e instale o Node.js no site Nodejs.org.

Como conectar-se ao Neptune usando o Node.js

1. Insira o seguinte para instalar o pacote `gremlin-javascript`:

   ```
   npm install gremlin
   ```

2. Crie um arquivo denominado `gremlinexample.js` e abra-o em um editor de texto.
3. Copie o seguinte no arquivo `gremlinexample.js`. Substitua *`your-neptune-endpoint`* pelo endereço da instância de banco de dados do Neptune.

   Para obter informações sobre como localizar o endereço da instância de banco de dados do Neptune, consulte a seção Conectar-se a endpoints do Amazon Neptune.

   ```
   const gremlin = require('gremlin');
   const DriverRemoteConnection = gremlin.driver.DriverRemoteConnection;
   const Graph = gremlin.structure.Graph;

   dc = new DriverRemoteConnection('wss://your-neptune-endpoint:8182/gremlin',{});

   const graph = new Graph();
   const g = graph.traversal().withRemote(dc);

   g.V().limit(1).count().next().
       then(data => {
           console.log(data);
           dc.close();
       }).catch(error => {
           console.log('ERROR', error);
           dc.close();
       });
   ```

4. Insira o seguinte comando para executar o exemplo:

```
node gremlinexample.js
```

O exemplo anterior retorna a contagem de um único vértice no gráfico usando a travessia `g.V().limit(1).count().next()`. Para consultar outro elemento, substitua-a por outra travessia do Gremlin com um dos métodos de término adequado.

> **Note**
>
> A parte final da consulta do Gremlin, `next()`, é necessária para enviar a travessia ao servidor para avaliação. Se você não incluir esse método ou outro método equivalente, a consulta não será enviada à instância de banco de dados do Neptune.

Os seguintes métodos enviam a consulta à instância de banco de dados do Neptune:

- `toList()`
- `toSet()`
- `next()`
- `nextTraverser()`
- `iterate()`

Use `next()` se precisar que os resultados da consulta sejam serializados e gerados, ou `iterate()` se não precisar.

> **Important**
>
> Este é um exemplo de Node.js independente. Se você planeja executar um código como esse em uma AWS Lambda função, consulte Exemplos de função do Lambda para obter detalhes sobre o uso JavaScript eficiente em uma função Lambda do Neptune.

Usar Go para conectar-se a uma instância de banco de dados do Neptune

Se você puder, sempre use a versão mais recente do cliente Apache TinkerPop Go Gremlin, gremlingo, compatível com a versão do seu motor. As versões mais recentes contêm várias correções de erros que melhoram a estabilidade, o desempenho e a usabilidade do cliente.

A `gremlingo` versão a ser usada normalmente se alinhará às TinkerPop versões descritas na tabela do cliente Java Gremlin.

> **Note**
>
> As versões 3.5.x do gremlingo são retrocompatíveis com as versões 3.4.x, desde que você use apenas os recursos TinkerPop 3.4.x nas consultas do Gremlin que você escreve.

A seção a seguir descreve a execução de um exemplo do Go que se conecta a uma instância de banco de dados do Amazon Neptune e executa um percurso do Gremlin.

Você deve seguir estas instruções em uma instância do Amazon EC2 na mesma nuvem privada virtual (VPC) que a instância de banco de dados do Neptune.

Antes de começar, faça o seguinte:

- Baixe e instale o Go 1.17 ou posterior no site gov.dev.

Como conectar-se ao Neptune usando Go

1. A partir de um diretório vazio, inicialize um novo módulo Go:

   ```
   go mod init example.com/gremlinExample
   ```

2. Adicione gremlin-go como uma dependência do novo módulo:

   ```
   go get github.com/apache/tinkerpop/gremlin-go/v3/driver
   ```

3. Crie um arquivo denominado `gremlinExample.go` e abra-o em um editor de texto.
4. Copie o seguinte no arquivo `gremlinExample.go`, substituindo *`(your neptune endpoint)`* pelo endereço da instância de banco de dados do Neptune:

   ```
   package main
   ```

```
import (
  "fmt"
  gremlingo "github.com/apache/tinkerpop/gremlin-go/v3/driver"
)

func main() {
  // Creating the connection to the server.
  driverRemoteConnection, err := gremlingo.NewDriverRemoteConnection("wss://(your
 neptune endpoint):8182/gremlin",
    func(settings *gremlingo.DriverRemoteConnectionSettings) {
      settings.TraversalSource = "g"
    })
  if err != nil {
    fmt.Println(err)
    return
  }
  // Cleanup
  defer driverRemoteConnection.Close()

  // Creating graph traversal
  g := gremlingo.Traversal_().WithRemote(driverRemoteConnection)

  // Perform traversal
  results, err := g.V().Limit(2).ToList()
  if err != nil {
    fmt.Println(err)
    return
  }
  // Print results
  for _, r := range results {
    fmt.Println(r.GetString())
  }
}
```

Note

No momento, o formato de certificado TLS do Neptune não é compatível com o Go 1.18+ com macOS e pode gerar um erro 509 ao tentar iniciar uma conexão. Para testes locais, isso pode ser ignorado adicionando “crypto/tls” às importações e modificando as configurações `DriverRemoteConnection` da seguinte forma:

```
// Creating the connection to the server.
driverRemoteConnection, err := gremlingo.NewDriverRemoteConnection("wss://
your-neptune-endpoint:8182/gremlin",
  func(settings *gremlingo.DriverRemoteConnectionSettings) {
      settings.TraversalSource = "g"
      settings.TlsConfig = &tls.Config{InsecureSkipVerify: true}
  })
```

5. Insira o seguinte comando para executar o exemplo:

```
go run gremlinExample.go
```

A consulta do Gremlin no final deste exemplo exibe os vértices (`g.V().Limit(2)`) em uma fatia. Essa fatia é então iterada e impressa com a função `fmt.Println` padrão.

> **Note**
>
> A parte final da consulta do Gremlin, `ToList()`, é necessária para enviar a travessia ao servidor para avaliação. Se você não incluir esse método ou outro método equivalente, a consulta não será enviada à instância de banco de dados do Neptune.

Os seguintes métodos enviam a consulta à instância de banco de dados do Neptune:

- `ToList()`
- `ToSet()`
- `Next()`
- `GetResultSet()`
- `Iterate()`

O exemplo anterior retorna os primeiros dois vértices do gráfico usando a travessia `g.V().Limit(2).ToList()`. Para consultar outro elemento, substitua-a por outra travessia do Gremlin com um dos métodos de término adequado.

Dicas de consulta do Gremlin

É possível usar as dicas de consulta para especificar estratégias de otimização e avaliação para uma consulta do Gremlin específica no Amazon Neptune.

Dicas de consulta são especificadas, adicionando uma `withSideEffect` etapa para a consulta com a seguinte sintaxe.

```
g.withSideEffect(hint, value)
```

- dica: identifica o tipo de dica a ser aplicada.
- value: determina o comportamento do aspecto do sistema em consideração.

Por exemplo, veja a seguir como incluir uma dica `repeatMode` em uma travessia Gremlin.

> **Note**
>
> Todas os efeitos colaterais de dicas de consulta do Gremlin são prefixados com `Neptune#`.

```
g.withSideEffect('Neptune#repeatMode',
 'DFS').V("3").repeat(out()).times(10).limit(1).path()
```

A consulta anterior instrui o mecanismo do Neptune a percorrer o grafo Profundidade primeiro (`DFS`) em vez do Neptune padrão, Amplitude primeiro (`BFS`).

As seções a seguir fornecem mais informações sobre as dicas de consulta disponíveis e o seu uso.

Tópicos

Dicas de consulta repeatMode do Gremlin

A dica de consulta `repeatMode` do Neptune especifica como o mecanismo do Neptune avalia a etapa `repeat()` em um percurso do Gremlin: primeiro em amplitude, primeiro em profundidade, ou profundidade em partes primeiro.

O modo de avaliação da etapa `repeat()` é importante quando ele é usado para encontrar ou seguir um caminho, em vez de simplesmente repetir uma etapa por um número limitado de vezes.

Sintaxe

A dica de consulta `repeatMode` é especificada, adicionando uma etapa `withSideEffect` para a consulta.

```
g.withSideEffect('Neptune#repeatMode', 'mode').gremlin-traversal
```

> **Note**
>
> Todas os efeitos colaterais de dicas de consulta do Gremlin são prefixados com `Neptune#`.

Modos disponíveis

- `BFS`

  Pesquisa primeiro em largura

  O modo de execução padrão para a `repeat()` etapa. Isso obtém todos os nós irmãos antes de se aprofundar ao longo do caminho.

  Esta versão é intensiva em memória e as fronteiras podem ficar muito grandes. Há um risco mais elevado de que a consulta seja executada sem memória e cancelada pelo mecanismo do Neptune. Isso é o que mais se aproxima de outras implementações do Gremlin.

- `DFS`

  Pesquisa primeiro em profundidade

  Acompanhe cada caminho até a profundidade máxima antes de prosseguir para a solução seguinte.

Isso usa menos memória. Pode fornecer melhor desempenho em situações como descobrir um único caminho desde um ponto de partida de vários saltos.

- `CHUNKED_DFS`

  Pesquisa primeiro em profundidade em partes

  Uma abordagem híbrida que explora o gráfico primeiro em profundidade em partes de 1.000 nós, em vez de 1 nó (`DFS`) ou todos os nós (`BFS`).

  O mecanismo do Neptune receberá até mil nós em cada nível antes de seguir o caminho mais profundo.

  Esta é uma abordagem equilibrada entre velocidade e o uso de memória.

  Também é útil se você deseja usar o `BFS`, mas a consulta estiver consumindo muita memória.

Exemplo

A seção a seguir descreve o efeito do modo de repetição em uma travessia do Gremlin.

No Neptune, modo padrão para a etapa `repeat()` é realizar uma estratégia de execução primeiro em amplitude (`BFS`) para todos os percursos.

Na maioria dos casos, a TinkerGraph implementação usa a mesma estratégia de execução, mas em alguns casos ela altera a execução de uma travessia.

Por exemplo, a TinkerGraph implementação modifica a consulta a seguir.

```
g.V("3").repeat(out()).times(10).limit(1).path()
```

A etapa `repeat()` nesta travessia é "desenrolada" na seguinte travessia, que resulta em uma estratégia primeiro em profundidade (`DFS`).

```
g.V(<id>).out().out().out().out().out().out().out().out().out().out().limit(1).path()
```

> **Important**
>
> O mecanismo de consulta do Neptune não faz isso automaticamente.

Breadth-first (BFS) é a estratégia de execução padrão e é semelhante à da maioria dos TinkerGraph casos. No entanto, existem certos casos em que as estratégias de primeiro em profundidade (DFS) são preferíveis.

BFS (padrão)

Primeiro em largura (BFS) é a estratégia de execução padrão para o operador `repeat()`.

```
g.V("3").repeat(out()).times(10).limit(1).path()
```

O mecanismo do Neptune explora totalmente as primeiras fronteiras de nove saltos antes de encontrar uma solução para dez saltos. Isso é eficaz em muitos casos, como consultas de caminho mais curto.

No entanto, no caso do exemplo anterior, a travessia seria mais rápida usando o modo primeiro em profundidade (DFS) para o operador `repeat()`.

DFS

A consulta a seguir usa o modo primeiro em profundidade (DFS) para o operador `repeat()`.

```
g.withSideEffect("Neptune#repeatMode", "DFS").V("3").repeat(out()).times(10).limit(1)
```

Isso segue cada solução individual para a profundidade máxima antes de explorar a solução seguinte.

Dicas de consulta noReordering do Gremlin

Ao enviar um percurso do Gremlin, o mecanismo de consulta do Neptune investiga a estrutura do percurso e reordena partes da consulta, tentando minimizar a quantidade de trabalho necessária para avaliação e tempo de resposta da consulta. Por exemplo, uma travessia com várias restrições, como várias etapas `has()`, normalmente não é avaliada na ordem determinada. Em vez de ser reordenado após a consulta, ele é verificado com análise estática.

O mecanismo de consulta do Neptune tenta identificar qual restrição é mais seletiva e executa essa primeiro. Isso muitas vezes ocasiona melhor desempenho, mas a ordem que o Neptune escolhe para avaliar a consulta pode não ser sempre a ideal.

Se você sabe exatamente as características dos dados e deseja ditar manualmente a ordem de execução da consulta, use a dica de consulta `noReordering` do Neptune para especificar que o percurso deve ser avaliado na ordem indicada.

Sintaxe

A dica de consulta `noReordering` é especificada, adicionando uma etapa `withSideEffect` para a consulta.

```
g.withSideEffect('Neptune#noReordering', true or false).gremlin-traversal
```

> **Note**
>
> Todas os efeitos colaterais de dicas de consulta do Gremlin são prefixados com `Neptune#`.

Valores disponíveis

- `true`
- `false`

Dica de consulta typePromotion do Gremlin

Quando você envia um percurso do Gremlin que filtra por um valor ou um intervalo numérico, o mecanismo de consulta do Neptune normalmente deve usar promoção de tipo ao executar a consulta. Isso significa que ele precisa examinar valores de todos os tipos que possam conter o valor que você está filtrando.

Por exemplo, se você estiver filtrando valores iguais a 55, o mecanismo deverá procurar números inteiros iguais a 55, números inteiros longos iguais a 55L, flutuantes iguais a 55,0, etc. Cada promoção de tipo exige uma pesquisa adicional no armazenamento, o que pode fazer com que uma consulta aparentemente simples leve um tempo inesperadamente longo para ser concluída.

Digamos que você esteja pesquisando todos os vértices com uma propriedade de idade do cliente maior que 5:

```
g.V().has('customerAge', gt(5))
```

Para executar esse percurso completamente, o Neptune deve expandir a consulta para examinar cada tipo numérico para o qual o valor que você está consultando poderia ser promovido. Nesse caso, o filtro `gt` deve ser aplicado para qualquer número inteiro acima de 5, qualquer valor longo acima de 5L, qualquer flutuante acima de 5,0 e qualquer dobro acima de 5,0. Como cada um desses tipos de promoção exige uma pesquisa adicional sobre armazenamento, você verá vários filtros por filtro numérico ao executar o API `profile` do Gremlin para essa consulta, e ela levará muito mais tempo para ser concluída do que o esperado.

Muitas vezes, a promoção de tipo é desnecessária porque você sabe de antemão que só precisa encontrar valores de um tipo específico. Nesse caso, você pode acelerar suas consultas drasticamente usando a dica de consulta `typePromotion` para desativar a promoção de tipos.

Sintaxe

A dica de consulta `typePromotion` é especificada, adicionando uma etapa `withSideEffect` para a consulta.

```
g.withSideEffect('Neptune#typePromotion', true or false).gremlin-traversal
```

> **Note**
>
> Todas os efeitos colaterais de dicas de consulta do Gremlin são prefixados com `Neptune#`.

Valores disponíveis

- `true`
- `false`

Para desativar a promoção de tipos para a consulta acima, você usaria:

```
g.withSideEffect('Neptune#typePromotion', false).V().has('customerAge', gt(5))
```

Dica de consulta useDFE do Gremlin

Use essa dica de consulta para permitir o uso do DFE para executar a consulta. Por padrão, o Neptune não usa o DFE sem que essa dica de consulta esteja definida como `true`, porque o parâmetro de instância neptune_dfe_query_engine tem como padrão `viaQueryHint`. Se você

definir esse parâmetro de instância como `enabled`, o mecanismo do DFE será usado para todas as consultas, exceto aquelas com a dica de consulta `useDFE` definida como `false`.

Exemplo de habilitação do DFE para uma consulta:

```
g.withSideEffect('Neptune#useDFE', true).V().out()
```

Dicas de consulta do Gremlin para usar o cache de resultados

As dicas de consulta a seguir podem ser usadas quando o cache de resultados da consulta está habilitado.

Dica de consulta **`enableResultCache`** do Gremlin

A dica de consulta `enableResultCache` com um valor de `true` fará com que os resultados da consulta sejam gerados do cache se já tiverem sido armazenados em cache. Caso contrário, ela gerará novos resultados e os armazena em cache até que sejam limpos do cache. Por exemplo: .

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

Posteriormente, é possível acessar os resultados em cache emitindo exatamente a mesma consulta novamente.

Se o valor dessa dica de consulta for `false`, ou se não estiver presente, os resultados da consulta não serão armazenados em cache. No entanto, configurá-lo como `false` não limpa os resultados existentes em cache. Para limpar os resultados em cache, use a dica `invalidateResultCache` ou `invalidateResultCachekey`.

Dica de consulta **`enableResultCacheWithTTL`** do Gremlin

A dica de consulta `enableResultCacheWithTTL` também gera resultados em cache, se houver, sem afetar o TTL dos resultados que já estão no cache. Se no momento não houver resultados em cache, a consulta gerará novos resultados e os armazenará em cache pelo tempo de vida (TTL) especificado pela dica de consulta `enableResultCacheWithTTL`. Esse tempo de vida é especificado em segundos. Por exemplo, a seguinte consulta especifica um tempo de vida de sessenta segundos:

```
g.with('Neptune#enableResultCacheWithTTL', 60)
 .V().has('genre','drama').in('likes')
```

Antes que os 60 segundos time-to-live terminem, você pode usar a mesma consulta (aquig`.V().has('genre','drama').in('likes')`) com a dica de consulta `enableResultCache` ou com a dica de `enableResultCacheWithTTL` consulta para acessar os resultados em cache.

> **Note**
>
> O tempo de vida especificado com `enableResultCacheWithTTL` não afeta os resultados já armazenados em cache.
>
> - Se os resultados tiverem sido previamente armazenados em cache usando `enableResultCache`, o cache deverá primeiro ser explicitamente limpo antes que o `enableResultCacheWithTTL` gere novos resultados e os armazene em cache para o TTL especificado.
> - Se os resultados tiverem sido previamente armazenados em cache usando `enableResultCachewithTTL`, o TTL anterior deverá primeiro expirar antes que o `enableResultCacheWithTTL` gere novos resultados e os armazene em cache para o TTL especificado.

Após o término do tempo de vida, os resultados em cache da consulta são apagados e uma instância subsequente da mesma consulta gera novos resultados. Se `enableResultCacheWithTTL` estiver associado à consulta subsequente, os novos resultados serão armazenados em cache com o TTL especificado.

Dica de consulta **`invalidateResultCacheKey`** do Gremlin

A dica de consulta `invalidateResultCacheKey` pode ter um valor `true` ou `false`. Um valor `true` faz com que os resultados em cache da consulta à qual `invalidateResultCacheKey` está associada sejam apagados. Por exemplo, o seguinte exemplo faz com que os resultados armazenados em cache para a chave de consulta `g.V().has('genre','drama').in('likes')` sejam apagados:

```
g.with('Neptune#invalidateResultCacheKey', true)
 .V().has('genre','drama').in('likes')
```

O exemplo de consulta acima não faz com que seus novos resultados sejam armazenados em cache. Será possível incluir `enableResultCache` (ou `enableResultCacheWithTTL`) na mesma

consulta se quiser armazenar em cache os novos resultados depois de limpar os existentes em cache:

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#invalidateResultCacheKey', true)
 .V().has('genre','drama').in('likes')
```

Dica de consulta **`invalidateResultCache`** do Gremlin

A dica de consulta `invalidateResultCache` pode ter um valor `true` ou `false`. Um valor `true` faz com que todos os resultados no cache de resultados sejam apagados. Por exemplo: .

```
g.with('Neptune#invalidateResultCache', true)
 .V().has('genre','drama').in('likes')
```

O exemplo de consulta acima não faz com que seus resultados sejam armazenados em cache. Será possível incluir `enableResultCache` (ou `enableResultCacheWithTTL`) na mesma consulta se quiser armazenar em cache os novos resultados depois de limpar completamente o cache existente:

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#invalidateResultCache', true)
 .V().has('genre','drama').in('likes')
```

Dica de consulta **`numResultsCached`** do Gremlin

A dica de consulta `numResultsCached` só pode ser usada com consultas que contenham `iterate()` e especifica o número máximo de resultados a serem armazenados em cache para a consulta à qual está associada. Observe que os resultados armazenados em cache quando `numResultsCached` está presente não são exibidos, apenas os armazenados em cache.

Por exemplo, a seguinte consulta especifica que até 100 dos resultados devem ser armazenados em cache, mas nenhum desses resultados deve ser gerado:

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#numResultsCached', 100)
 .V().has('genre','drama').in('likes').iterate()
```

Depois, você pode usar uma consulta como a seguinte para recuperar uma variedade de resultados em cache (aqui, os dez primeiros):

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#numResultsCached', 100)
 .V().has('genre','drama').in('likes').range(0, 10)
```

Dica de consulta **`noCacheExceptions`** do Gremlin

A dica de consulta `noCacheExceptions` pode ter um valor `true` ou `false`. Um valor `true` faz com que todas as exceções relacionadas ao cache de resultados sejam suprimidas. Por exemplo: .

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#noCacheExceptions', true)
 .V().has('genre','drama').in('likes')
```

Especificamente, isso suprime a `QueryLimitExceededException`, que será gerada se os resultados de uma consulta forem grandes demais para caber no cache de resultados.

API de status de consulta do Gremlin

Para obter o status das consultas do Gremlin, use HTTP `GET` ou `POST` para fazer uma solicitação ao endpoint de `https://`*`your-neptune-endpoint`*`:`*`port`*`/gremlin/status`.

Parâmetros de solicitação do status de consulta do Gremlin

- queryId (opcional): o ID de uma consulta do Gremlin em execução. Exibe apenas o status da consulta especificada.
- includeWaiting (opcional): exibe o status de todas as consultas em espera.

  Normalmente, somente as consultas em execução são incluídas na resposta, mas quando o parâmetro `includeWaiting` é especificado, o status de todas as consultas em espera também é exibido.

Sintaxe da resposta do status de consulta do Gremlin

```
{
  "acceptedQueryCount": integer,
  "runningQueryCount": integer,
  "queries": [
    {
      "queryId":"guid",
```

```
      "queryEvalStats":
        {
          "waited": integer,
          "elapsed": integer,
          "cancelled": boolean
        },
      "queryString": "string"
    }
  ]
}
```

Valores da resposta do status de consulta do Gremlin

- aceito QueryCount — O número de consultas que foram aceitas, mas ainda não concluídas, incluindo consultas na fila.
- em execução QueryCount — O número de consultas do Gremlin atualmente em execução.
- queries: uma lista de consultas do Gremlin atuais.
- queryId: um ID de GUID para a consulta. O Neptune atribui automaticamente esse valor de ID a cada consulta, ou você também pode atribuir seu próprio ID (consulte Injetar um ID personalizado em uma consulta do Gremlin ou do SPARQL no Neptune).
- query EvalStats — Estatísticas dessa consulta.
- subqueries: o número de subconsultas nessa consulta.
- elapsed: o número de milissegundos em que a consulta esteve em execução até o momento.
- cancelled: verdadeiro indica que a consulta foi cancelada.
- queryString: a consulta enviada. Ela será truncada para 1024 caracteres se for maior do que isso.
- waited: indica quanto tempo a consulta esperou, em milissegundos.

Exemplo de status de consulta do Gremlin

A seguir está um exemplo de comando de status usando `curl` e HTTP `GET`.

```
curl https://your-neptune-endpoint:port/gremlin/status
```

Essa saída mostra uma única consulta em execução.

```
{
  "acceptedQueryCount":9,
```

```
  "runningQueryCount":1,
  "queries": [
    {
      "queryId":"fb34cd3e-f37c-4d12-9cf2-03bb741bf54f",
      "queryEvalStats":
        {
          "waited": 0,
          "elapsed": 23,
          "cancelled": false
        },
      "queryString": "g.V().out().count()"
    }
  ]
}
```

Cancelamento de consultas do Gremlin

Para obter o status das consultas do Gremlin, use HTTP `GET` ou `POST` para fazer uma solicitação ao endpoint de `https://`*`your-neptune-endpoint`*`:`*`port`*`/gremlin/status`.

Parâmetros de solicitação do cancelamento de consultas do Gremlin

- cancelQuery: obrigatório para o cancelamento. Esse parâmetro não tem valor correspondente.
- queryId: o ID da consulta do Gremlin em execução a ser cancelada.

Exemplo de cancelamento de consultas do Gremlin

Veja a seguir um exemplo do comando `curl` para cancelar uma consulta.

```
curl https://your-neptune-endpoint:port/gremlin/status \
  --data-urlencode "cancelQuery" \
  --data-urlencode "queryId=fb34cd3e-f37c-4d12-9cf2-03bb741bf54f"
```

Cancelamento bem-sucedido retorna HTTP `200` OK.

Suporte para sessões baseadas em script do Gremlin

É possível usar sessões do Gremlin com transações implícitas no Amazon Neptune. Para obter informações sobre as sessões do Gremlin, consulte Considerando as sessões na documentação do TinkerPop Apache. As seções abaixo descrevem como usar sessões do Gremlin com Java.

> **Note**
>
> Esse atributo está disponível a partir da versão 1.0.1.0.200463.0 do mecanismo do Neptune. Começando com a versão 1.1.1.0 TinkerPop e a versão 3.5.2 do motor Neptune, você também pode usar. Transações do Gremlin

> **Important**
>
> No momento, o tempo máximo que o Neptune pode manter uma sessão aberta baseada em script é dez minutos. Se você não fechar uma sessão antes desse tempo, a sessão atingirá o tempo limite e tudo associado a ela será revertido.

Tópicos

- Sessões do Gremlin no console do Gremlin
- Sessões do Gremlin no Gremlin Language Variant

Sessões do Gremlin no console do Gremlin

Se você criar uma conexão remota no Console do Gremlin sem o parâmetro `session`, a conexão remota será criada no modo sem sessão. Nesse modo, cada solicitação enviada ao servidor é tratada como uma transação completa por si só, e nenhum estado é salvo entre as solicitações. Se uma solicitação falhar, apenas essa solicitação será revertida.

Se você criar uma conexão remota que não use o parâmetro `session`, criará uma sessão baseada em script que durará até que você feche a conexão remota. Cada sessão é identificada por um UUID exclusivo que o console gera e retorna para você.

Veja a seguir um exemplo de uma chamada de console que cria uma sessão. Depois que as consultas são enviadas, outra chamada fecha a sessão e confirma as consultas.

> **Note**
>
> O cliente do Gremlin deve estar sempre fechado para liberar recursos do lado do servidor.

```
gremlin> :remote connect tinkerpop.server conf/neptune-remote.yaml session
```

```
  . . .
  . . .
gremlin> :remote close
```

Para obter mais informações e exemplos, consulte Sessões na TinkerPop documentação.

Todas as consultas executadas durante uma sessão formam uma única transação que não é confirmada até que todas as consultas sejam bem-sucedidas e você feche a conexão remota. Se uma consulta falhar, ou se você não fechar a conexão no tempo de vida máximo da sessão permitido pelo Neptune, a transação da sessão não será confirmada e todas as consultas associadas serão revertidas.

Sessões do Gremlin no Gremlin Language Variant

No Gremlin Language Variant (GLV), você precisa criar um objeto `SessionedClient` para executar várias consultas em uma única transação, como no exemplo a seguir.

```
try {                              // line 1
  Cluster cluster = Cluster.open();                    // line 2
  Client client = cluster.connect("sessionName");      // line 3
   ...
   ...
} finally {
  // Always close. If there are no errors, the transaction is committed; otherwise,
 it's rolled back.
  client.close();
}
```

A linha 3 no exemplo anterior cria o objeto `SessionedClient` de acordo com as opções de configuração definidas para o cluster em questão. A string *`sessionName`* que você passar para o método de conexão se torna o nome exclusivo da sessão. Para evitar colisões, use um UUID para o nome.

O cliente inicia uma transação de sessão quando é inicializado. Todas as consultas executadas durante o formulário de sessão são confirmadas somente quando você chama `client.close( )`. Novamente, se uma consulta falhar, ou se você não fechar a conexão no tempo de vida máximo da sessão permitido pelo Neptune, a transação da sessão falhará e todas as consultas associadas serão revertidas.

> **Note**
>
> O cliente do Gremlin deve estar sempre fechado para liberar recursos do lado do servidor.

```
GraphTraversalSource g = traversal().withRemote(conn);

Transaction tx = g.tx();

// Spawn a GraphTraversalSource from the Transaction.
// Traversals spawned from gtx are executed within a single transaction.
GraphTraversalSource gtx = tx.begin();
try {
  gtx.addV('person').iterate();
  gtx.addV('software').iterate();

  tx.commit();
} finally {
    if (tx.isOpen()) {
        tx.rollback();
    }
}
```

Transações do Gremlin no Neptune

Existem vários contextos nos quais as transações do Gremlin são executadas. Ao trabalhar com o Gremlin, é importante entender o contexto em que você está trabalhando e quais são suas implicações:

- **`Script-based`**: as solicitações são feitas usando strings do Gremlin baseadas em texto, como esta:
  - Usar o driver Java e `Client.submit(`*`string`*`)`.
  - Usar o console do Gremlin e `:remote connect`.
  - Usar a API HTTP.
- **`Bytecode-based`**: as solicitações são feitas usando o bytecode do Gremlin serializado típico das Variantes da linguagem Gremlin (GLV).

  Por exemplo, usando o driver Java, `g = traversal().withRemote(...)`.

Para qualquer um dos contextos acima, há o contexto adicional da solicitação sendo enviada sem sessão ou vinculada a uma sessão.

> **Note**
>
> As transações do Gremlin sempre devem ser confirmadas ou revertidas, para que os recursos do lado do servidor possam ser liberados.

Solicitações sem sessão

Quando não há sessão, uma solicitação equivale a uma única transação.

Para scripts, a implicação é que uma ou mais declarações do Gremlin enviadas em uma única solicitação serão confirmadas ou revertidas como uma única transação. Por exemplo: .

```
Cluster cluster = Cluster.open();
Client client = cluster.connect(); // sessionless
// 3 vertex additions in one request/transaction:
client.submit("g.addV();g.addV();g.addV()").all().get();
```

Para bytecode, uma solicitação sem sessão é feita para cada percurso gerado e executado a partir de g:

```
GraphTraversalSource g = traversal().withRemote(...);

// 3 vertex additions in three individual requests/transactions:
g.addV().iterate();
g.addV().iterate();
g.addV().iterate();

// 3 vertex additions in one single request/transaction:
g.addV().addV().addV().iterate();
```

Solicitações vinculadas a uma sessão

Quando vinculadas a uma sessão, várias solicitações podem ser aplicadas no contexto de uma única transação.

Para scripts, a implicação é que não há necessidade de concatenar todas as operações do grafo em um único valor de string incorporado:

```
Cluster cluster = Cluster.open();
Client client = cluster.connect(sessionName); // session
try {
    // 3 vertex additions in one request/transaction:
    client.submit("g.addV();g.addV();g.addV()").all().get();
} finally {
    client.close();
}

try {
    // 3 vertex additions in three requests, but one transaction:
    client.submit("g.addV()").all().get(); // starts a new transaction with the same
 sessionName
    client.submit("g.addV()").all().get();
    client.submit("g.addV()").all().get();
} finally {
    client.close();
}
```

Para o bytecode, depois TinkerPop `3.5.x`, a transação pode ser controlada explicitamente e a sessão gerenciada de forma transparente. As Variantes da linguagem Gremlin (GLV) são compatíveis com a sintaxe `tx()` do Gremlin para `commit()` ou `rollback()` uma transação da seguinte forma:

```
GraphTraversalSource g = traversal().withRemote(conn);

Transaction tx = g.tx();

// Spawn a GraphTraversalSource from the Transaction.
// Traversals spawned from gtx are executed within a single transaction.
GraphTraversalSource gtx = tx.begin();
try {
    gtx.addV('person').iterate();
    gtx.addV('software').iterate();

    tx.commit();
} finally {
    if (tx.isOpen()) {
        tx.rollback();
    }
}
```

Embora o exemplo acima esteja escrito em Java, você também pode usar essa sintaxe `tx()` em Python, Javascript e .NET.

> **Warning**
>
> As consultas somente leitura sem sessão são executadas em isolamento SNAPSHOT, mas as consultas somente leitura executadas em uma transação explícita são executadas em isolamento SERIALIZABLE. As consultas somente leitura executadas em isolamento `SERIALIZABLE` geram maior sobrecarga e podem bloquear ou bloqueadas por gravações simultâneas, ao contrário das executadas em isolamento `SNAPSHOT`.

Usar a API do Gremlin com o Amazon Neptune

> **Note**
>
> O Amazon Neptune não é compatível com a propriedade `bindings`.

Todas as solicitações HTTPS usam um único endpoint: `https://`*`your-neptune-endpoint`*`:`*`port`*`/gremlin`. Todas as conexões do Neptune devem usar HTTPS.

Você pode conectar o Gremlin Console a um gráfico de Neptune diretamente por meio de. WebSockets

Para obter mais informações sobre como se conectar ao endpoint do Gremlin, consulte Acessar o grafo do Neptune com o Gremlin.

A implementação do Gremlin no Amazon Neptune tem detalhes específicos e diferenças que você precisa considerar. Para ter mais informações, consulte Conformidade com os padrões do Gremlin no Amazon Neptune.

Para obter informações sobre a linguagem Gremlin e as travessias, consulte The Traversal na documentação do Apache. TinkerPop

Armazenar em cache os resultados da consulta no Gremlin do Amazon Neptune

A partir da versão 1.0.5.1 do mecanismo, o Amazon Neptune é compatível com um cache de resultados para consultas do Gremlin.

É possível habilitar o cache de resultados da consulta e, depois, usar uma dica de consulta para armazenar em cache os resultados de uma consulta somente leitura do Gremlin.

Depois, qualquer nova execução da consulta recupera os resultados em cache com baixa latência e sem custos de E/S, desde que eles ainda estejam no cache. Isso funciona para consultas enviadas em um endpoint HTTP e usando Websockets, como código de bytes ou em formato de string.

> **Note**
>
> As consultas enviadas ao endpoint do perfil não são armazenadas em cache, mesmo quando o cache de consultas está habilitado.

É possível controlar como o cache de resultados da consulta do Neptune se comporta de várias maneiras. Por exemplo: .

- Você pode obter resultados em cache paginados, em blocos.
- Você pode especificar o time-to-live (TTL) para consultas específicas.
- Você pode limpar o cache para consultas específicas.
- É possível limpar todo o cache.
- Você pode configurar para ser notificado se os resultados excederem o tamanho do cache.

O cache é mantido usando uma política least-recently-used (LRU), o que significa que, quando o espaço alocado para o cache estiver cheio, os least-recently-used resultados serão removidos para liberar espaço quando novos resultados forem armazenados em cache.

> **Important**
>
> O cache de resultados da consulta não está disponível em nossos tipos de instância `t3.medium` ou `t4.medium`.

Habilitar o cache de resultados da consulta no Neptune

Para habilitar o cache de resultados de consultas no Neptune, use o console para definir o parâmetro de instância de banco de dados `neptune_result_cache` como 1 (habilitado).

Depois que o cache de resultados é habilitado, o Neptune reserva uma parte da memória atual para armazenar em cache os resultados das consultas. Quanto maior o tipo de instância que você estiver usando e quanto mais memória estiver disponível, mais memória o Neptune reservará para o cache.

Se a memória cache dos resultados ficar cheia, o Neptune least-recently-used descarta automaticamente os resultados em cache (LRU) para dar lugar a novos.

Você pode conferir o status atual do cache de resultados usando o comando Status de instância.

Usar dicas para armazenar em cache os resultados da consulta

Depois que o cache de resultados da consulta estiver habilitado, você usará dicas de consulta para controlar o armazenamento de consultas em cache. Todos os exemplos abaixo se aplicam ao mesmo percurso de consulta, a saber:

```
g.V().has('genre','drama').in('likes')
```

Utilizar o **`enableResultCache`**

Com o cache de resultados de consultas habilitado, é possível armazenar em cache os resultados de uma consulta do Gremlin usando a dica de consulta `enableResultCache`, da seguinte forma:

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

Depois, o Neptune exibe os resultados da consulta e também os armazena em cache. Posteriormente, é possível acessar os resultados em cache emitindo exatamente a mesma consulta novamente.

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

A chave de cache que identifica os resultados em cache é a própria string da consulta, a saber:

```
g.V().has('genre','drama').in('likes')
```

Utilizar o `enableResultCacheWithTTL`

É possível especificar por quanto tempo os resultados da consulta devem ser armazenados em cache usando a dica de consulta `enableResultCacheWithTTL`. Por exemplo, a seguinte consulta especifica que os resultados da consulta devem expirar após 120 segundos:

```
g.with('Neptune#enableResultCacheWithTTL', 120)
 .V().has('genre','drama').in('likes')
```

Novamente, a chave de cache que identifica os resultados em cache é a própria string da consulta:

```
g.V().has('genre','drama').in('likes')
```

E, novamente, é possível acessar os resultados em cache usando essa string de consulta com a dica de consulta `enableResultCache`:

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

Se tiverem passado 120 segundos ou mais desde que os resultados foram armazenados em cache, essa consulta retornará novos resultados e os armazenará em cache, sem nenhum time-to-live.

Você também pode acessar os resultados em cache emitindo a mesma consulta novamente com a dica de consulta `enableResultCacheWithTTL`. Por exemplo: .

```
g.with('Neptune#enableResultCacheWithTTL', 140)
 .V().has('genre','drama').in('likes')
```

Até transcorrerem 120 segundos (ou seja, o TTL em vigor no momento), essa nova consulta que usa a dica de consulta `enableResultCacheWithTTL` gera os resultados em cache. Após 120 segundos, ele retornaria novos resultados e os armazenaria em time-to-live cache com 140 segundos.

> **Note**
>
> Se os resultados de uma chave de consulta já estiverem armazenados em cache, a mesma chave de consulta `enableResultCacheWithTTL` não gerará novos resultados e não terá efeito sobre os time-to-live resultados atualmente armazenados em cache.

- Se os resultados tiverem sido previamente armazenados em cache usando `enableResultCache`, o cache deverá primeiro ser limpo antes que o `enableResultCacheWithTTL` gere novos resultados e os armazene em cache para o TTL especificado.
- Se os resultados tiverem sido previamente armazenados em cache usando `enableResultCachewithTTL`, o TTL anterior deverá primeiro expirar antes que o `enableResultCacheWithTTL` gere novos resultados e os armazene em cache para o TTL especificado.

Utilizar o **`invalidateResultCacheKey`**

É possível usar a dica de consulta `invalidateResultCacheKey` para limpar os resultados em cache de uma consulta específica. Por exemplo: .

```
g.with('Neptune#invalidateResultCacheKey', true)
 .V().has('genre','drama').in('likes')
```

Essa consulta limpa o cache da chave de consulta, `g.V().has('genre','drama').in('likes')`, e gera novos resultados para essa consulta.

Você também pode combinar `invalidateResultCacheKey` com `enableResultCache` ou `enableResultCacheWithTTL`. Por exemplo, a seguinte consulta limpa os resultados atuais em cache, armazena novos resultados em cache e os exibe:

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#invalidateResultCacheKey', true)
 .V().has('genre','drama').in('likes')
```

Utilizar o **`invalidateResultCache`**

Você pode usar a dica de consulta `invalidateResultCache` para limpar todos os resultados em cache no cache de resultados da consulta. Por exemplo: .

```
g.with('Neptune#invalidateResultCache', true)
 .V().has('genre','drama').in('likes')
```

Essa consulta limpa todo o cache de resultados e gera novos resultados para essa consulta.

Você também pode combinar `invalidateResultCache` com `enableResultCache` ou `enableResultCacheWithTTL`. Por exemplo, a seguinte consulta limpa todo o cache de resultados, armazena novos resultados em cache para essa consulta e os exibe:

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#invalidateResultCache', true)
 .V().has('genre','drama').in('likes')
```

Paginar resultados de consulta em cache

Suponha que você já tenha armazenado em cache um grande número de resultados como este:

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

Agora, suponha que você emita a seguinte consulta de intervalo:

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes').range(0,10)
```

O Neptune primeiro procura a chave de cache completa, ou seja, `g.V().has('genre','drama').in('likes').range(0,10)`. Se essa chave não existir, o Neptune, depois, vai conferir se há uma chave para essa string de consulta sem o intervalo (ou seja, `g.V().has('genre','drama').in('likes')`). Quando encontra essa chave, o Neptune busca os dez primeiros resultados do cache, conforme especifica o intervalo.

> **Note**
>
> Se você usar a dica de consulta `invalidateResultCacheKey` com uma consulta que tenha um intervalo no final, o Neptune limpará o cache de uma consulta sem o intervalo se não encontrar uma correspondência exata para a consulta com o intervalo.

Usar o **`numResultsCached`** com o **`.iterate()`**

Usando a dica de consulta `numResultsCached`, você pode preencher o cache de resultados sem exibir todos os resultados que estão sendo armazenados em cache, o que pode ser útil quando você prefere paginar um grande número de resultados.

A dica de consulta `numResultsCached` só funciona com consultas que terminam com `iterate()`.

Por exemplo, se você quiser armazenar em cache os primeiros 50 resultados da consulta de amostra:

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 50)
 .V().has('genre','drama').in('likes').iterate()
```

Nesse caso, a chave de consulta no cache é: `g.with("Neptune#numResultsCached", 50).V().has('genre','drama').in('likes')`. Agora você pode recuperar os dez primeiros resultados em cache com esta consulta:

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 50)
 .V().has('genre','drama').in('likes').range(0, 10)
```

Além disso, é possível recuperar os próximos dez resultados da consulta da seguinte forma:

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 50)
 .V().has('genre','drama').in('likes').range(10, 20)
```

Não se esqueça de incluir a dica `numResultsCached`! É uma parte essencial da chave de consulta e, portanto, deve estar presente para acessar os resultados em cache.

Ao usar **`numResultsCached`**, tenha em mente que:

- O número fornecido com **`numResultsCached`** é aplicado ao final da consulta. Isso significa, por exemplo, que a seguinte consulta realmente armazena em cache os resultados no intervalo `(1000, 1500)`:

  ```
  g.with("Neptune#enableResultCache", true)
   .with("Neptune#numResultsCached", 500)
   .V().range(1000, 2000).iterate()
  ```

- O número fornecido com **`numResultsCached`** especifica o número máximo de resultados a serem armazenados em cache. Isso significa, por exemplo, que a seguinte consulta realmente armazena em cache os resultados no intervalo `(1000, 2000)`:

  ```
  g.with("Neptune#enableResultCache", true)
  ```

```
 .with("Neptune#numResultsCached", 100000)
 .V().range(1000, 2000).iterate()
```

- Os resultados armazenados em cache por consultas que terminam com **`.range().iterate()`** têm o próprio intervalo. Por exemplo, suponhamos que você armazene os resultados em cache usando uma consulta como esta:

  ```
  g.with("Neptune#enableResultCache", true)
   .with("Neptune#numResultsCached", 500)
   .V().range(1000, 2000).iterate()
  ```

  Para recuperar os primeiros cem resultados do cache, você escreveria uma consulta como esta:

  ```
  g.with("Neptune#enableResultCache", true)
   .with("Neptune#numResultsCached", 500)
   .V().range(1000, 2000).range(0, 100)
  ```

  Esses cem resultados seriam equivalentes aos resultados da consulta básica no intervalo `(1000, 1100)`.

As chaves de cache de consulta usadas para localizar resultados em cache

Depois que os resultados de uma consulta são armazenados em cache, as consultas subsequentes com a mesma chave de cache de consulta recuperam os resultados do cache em vez de gerar novos. A chave de cache de uma consulta é avaliada da seguinte forma:

1. Todas as dicas de consulta relacionadas ao cache são ignoradas, exceto `numResultsCached`.
2. Uma etapa final `iterate()` é ignorada.
3. O restante da consulta é ordenado de acordo com a representação em código de bytes.

A string resultante é comparada com um índice dos resultados da consulta que já estão no cache para determinar se há uma ocorrência de cache para a consulta.

Por exemplo, tome esta consulta como exemplo:

```
g.withSideEffect('Neptune#typePromotion', false).with("Neptune#enableResultCache",
 true)
 .with("Neptune#numResultsCached", 50)
```

```
  .V().has('genre','drama').in('likes').iterate()
```

Ela será armazenada como a versão em código de bytes desta:

```
g.withSideEffect('Neptune#typePromotion', false)
 .with("Neptune#numResultsCached", 50)
 .V().has('genre','drama').in('likes')
```

Exceções relacionadas ao cache de resultados

Se os resultados de uma consulta que você está tentando armazenar em cache forem muito grandes para caber na memória de cache, mesmo depois de remover tudo o que estava anteriormente armazenado, o Neptune gerará uma falha `QueryLimitExceededException`. Nenhum resultado é gerado, e a exceção gera a seguinte mensagem de erro:

```
The result size is larger than the allocated cache,
      please refer to results cache best practices for options to rerun the query.
```

É possível suprimir essa mensagem usando a dica de consulta `noCacheExceptions`, da seguinte forma:

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#noCacheExceptions', true)
 .V().has('genre','drama').in('likes')
```

Criar surtos eficientes com as etapas **`mergeV()`** e **`mergeE()`** do Gremlin.

Um upsert (ou inserção condicional) reutilizará um vértice ou uma borda se já existir ou criará um desses elementos se não existir. Upserts eficientes podem fazer uma diferença significativa no desempenho das consultas do Gremlin.

Os upserts permitem que você escreva operações de inserção idempotentes: não importa quantas vezes você execute essa operação, o resultado geral é o mesmo. Isso é útil em cenários de gravação altamente simultâneos em que modificações simultâneas na mesma parte do grafo podem forçar a reversão de uma ou mais transações com `ConcurrentModificationException`, exigindo novas tentativas.

Por exemplo, a consulta a seguir inverte um vértice usando o `Map` fornecido para primeiro tentar encontrar um vértice com um `T.id` de `"v-1"`. Se esse vértice for encontrado, ele será gerado.

Se não for encontrado, um vértice com esse `id` e a propriedade será criado por meio da cláusula `onCreate`.

```
g.mergeV([(id):'v-1']).
  option(onCreate, [(label): 'PERSON', 'email': 'person-1@example.org'])
```

Agrupar upserts em lote para melhorar o throughput

Para cenários de gravação de throughput, é possível encadear as etapas `mergeV()` e `mergeE()` para aplicar upserts em vértices e bordas em lote. O agrupamento em lote reduz a sobrecarga transacional de aplicar upserts a um grande número de vértices e bordas. Depois, é possível melhorar ainda mais o throughput aplicando upserts às solicitações em lote paralelamente usando vários clientes.

Como regra, recomendamos aplicar upserts a cerca de duzentos registros por solicitação em lote. Registro é um único rótulo ou propriedade de vértice ou borda. Um vértice com um único rótulo e quatro propriedades, por exemplo, cria cinco registros. Uma borda com um rótulo e uma única propriedade cria dois registros. Se você quiser aplicar upserts a lotes de vértices, cada um com um único rótulo e quatro propriedades, deverá começar com um tamanho de lote de quarenta, porque `200 / (1 + 4) = 40`.

É possível experimentar o tamanho do lote. Um valor de duzentos registros por lote é um bom ponto de partida, mas o tamanho ideal pode ser maior ou menor, dependendo da workload. Observe, no entanto, que o Neptune pode limitar o número total de etapas do Gremlin por solicitação. Esse limite não está documentado, mas, por segurança, tente garantir que suas solicitações não contenham mais de 1.500 etapas do Gremlin. O Neptune pode rejeitar grandes solicitações em lote com mais de 1.500 etapas.

Para aumentar o throughput, é possível inverter lotes em paralelo usando vários clientes (consulte Criação de gravações eficientes com multi-thread do Gremlin). O número de clientes deve ser igual ao número de threads de operador em sua instância de gravador do Neptune, que normalmente é duas vezes o número de vCPUs no servidor. Por exemplo, uma instância `r5.8xlarge` tem 32 vCPUs e 64 threads de operador. Para cenários de gravação de throughput usando um `r5.8xlarge`, você usaria 64 clientes gravando upserts em lote no Neptune em paralelo.

Cada cliente deve enviar uma solicitação em lote e aguardar a conclusão da solicitação antes de enviar outra solicitação. Embora os vários clientes funcionem paralelamente, cada cliente individual envia solicitações em série. Isso garante que o servidor receba um fluxo constante de solicitações

que ocupem todos os threads de operador sem inundar a fila de solicitações do lado do servidor (consulte Dimensionar instâncias de banco de dados em um cluster de banco de dados do Neptune).

Tentar evitar etapas que gerem vários percursos

Quando uma etapa do Gremlin é executada, ela pega um percurso de entrada e emite um ou mais percursos de saída. O número de percursos emitidos por uma etapa determina o número de vezes que a próxima etapa é executada.

Normalmente, ao realizar operações em lote, é recomendável que cada operação, como o vértice de upsert A, seja executada uma vez, para que a sequência de operações seja a seguinte: vértice de upsert A, depois vértice de upsert B, vértice de upsert C, etc. Desde que uma etapa crie ou modifique somente um elemento, ela emite somente um percurso, e as etapas que representam a próxima operação serão executadas somente uma vez. Se, por outro lado, uma operação criar ou modificar mais de um elemento, ela emitirá vários percursos o que, por sua vez, faz com que as etapas subsequentes sejam executadas várias vezes, uma vez por percurso emitido. Isso pode fazer com que o banco de dados execute trabalho adicional desnecessário e, em alguns casos, pode ocasionar a criação de vértices, bordas ou valores de propriedades adicionais indesejados.

Um exemplo de como as coisas podem dar errado é uma consulta como `g.V().addV()`. Essa consulta simples adiciona um vértice para cada vértice encontrado no grafo, porque `V()` emite um percurso para cada vértice no grafo e cada um desses percursos aciona uma chamada para `addV()`.

Consulte Misturar upserts e inserções para saber como lidar com operações que podem emitir vários percursos.

Aplicar upserts a vértices

A etapa `mergeV()` foi projetada especificamente para aplicar upserts a vértices. Ela usa como argumento um `Map` que representa elementos correspondentes aos vértices existentes no grafo e, se algum elemento não for encontrado, usará esse `Map` para criar um vértice. A etapa também permite alterar o comportamento no caso de uma criação ou uma correspondência, em que o modulador `option()` pode ser aplicado com tokens `Merge.onCreate` e `Merge.onMatch` para controlar esses respectivos comportamentos. Consulte a documentação de TinkerPop referência para obter mais informações sobre como usar essa etapa.

É possível usar um ID de vértice para determinar se existe um vértice específico. Essa é a abordagem preferencial, porque o Neptune otimiza upserts para casos de uso altamente simultâneos

em torno de IDs. Por exemplo, a seguinte consulta criará um vértice com um ID específico, se ele ainda não existir, ou o reutilizará se existir:

```
g.mergeV([(T.id): 'v-1']).
    option(onCreate, [(T.label): 'PERSON', email: 'person-1@example.org', age: 21]).
    option(onMatch, [age: 22]).
  id()
```

Observe que essa consulta termina com uma etapa `id()`. Embora não seja estritamente necessário para aplicar upserts ao vértice, uma etapa `id()` até o final de uma consulta de upsert garante que o servidor não serialize todas as propriedades do vértice de volta para o cliente, o que ajuda a reduzir o custo de bloqueio da consulta.

Você também pode usar uma propriedade de vértice para identificar um vértice:

```
g.mergeV([email: 'person-1@example.org']).
    option(onCreate, [(T.label): 'PERSON', age: 21]).
    option(onMatch, [age: 22]).
  id()
```

Se possível, use seus próprios IDs fornecidos pelo usuário para criar vértices e use esses IDs para determinar se existe um vértice durante uma operação de upsert. Isso permite que o Neptune otimize os upserts. Um upsert baseado em ID pode ser significativamente mais eficiente do que um upsert baseado em propriedade quando modificações simultâneas são comuns.

Encadear upserts de vértice

É possível encadear upserts de vértice para inseri-los em um lote:

```
g.V('v-1')
 .fold()
 .coalesce(unfold(),
          addV('Person').property(id, 'v-1')
                        .property('email', 'person-1@example.org'))
 .V('v-2')
 .fold()
 .coalesce(unfold(),
          addV('Person').property(id, 'v-2')
                        .property('email', 'person-2@example.org'))
 .V('v-3')
 .fold()
```

```
  .coalesce(unfold(),
            addV('Person').property(id, 'v-3')
                          .property('email', 'person-3@example.org'))
  .id()
```

Você também pode usar esta sintaxe `mergeV()`:

```
g.mergeV([(T.id): 'v-1', (T.label): 'PERSON', email: 'person-1@example.org']).
  mergeV([(T.id): 'v-2', (T.label): 'PERSON', email: 'person-2@example.org']).
  mergeV([(T.id): 'v-3', (T.label): 'PERSON', email: 'person-3@example.org'])
```

No entanto, como essa forma de consulta inclui elementos nos critérios de pesquisa que são supérfluos para a pesquisa básica por `id`, ela não é tão eficiente quanto a consulta anterior.

Aplicar upserts a bordas

A etapa `mergeE()` foi projetada especificamente para aplicar upserts a bordas. Ela usa como argumento um `Map` que representa elementos correspondentes às bordas existentes no grafo e, se algum elemento não for encontrado, usará esse `Map` para criar uma borda. A etapa também permite alterar o comportamento no caso de uma criação ou uma correspondência, em que o modulador `option()` pode ser aplicado com tokens `Merge.onCreate` e `Merge.onMatch` para controlar esses respectivos comportamentos. Consulte a documentação de TinkerPop referência para obter mais informações sobre como usar essa etapa.

É possível usar IDs de borda para aplicar upserts a bordas da mesma forma que a vértices usando IDs de vértice personalizados. Novamente, essa é a abordagem preferencial porque permite que o Neptune otimize a consulta. Por exemplo, a consulta a seguir cria uma borda com base em seu ID de borda, se ela ainda não existir, ou a reutiliza se existir. A consulta também usará os IDs dos vértices `Direction.from` e `Direction.to` se precisar criar uma borda:

```
g.mergeE([(T.id): 'e-1']).
    option(onCreate, [(from): 'v-1', (to): 'v-2', weight: 1.0]).
    option(onMatch, [weight: 0.5]).
  id()
```

Observe que essa consulta termina com uma etapa `id()`. Embora não seja estritamente necessário para aplicar upserts à borda, adicionar uma etapa `id()` até o final de uma consulta de upsert garante que o servidor não serialize todas as propriedades da borda de volta para o cliente, o que ajuda a reduzir o custo de bloqueio da consulta.

Muitas aplicações usam IDs de vértice personalizados, mas deixam o Neptune gerar IDs de borda. Se você não sabe o ID de uma borda, mas sim os IDs de vértice `from` e `to`, pode usar esse tipo de consulta para aplicar upsert a uma borda:

```
g.mergeE([(from): 'v-1', (to): 'v-2', (T.label): 'KNOWS']).
  id()
```

Todos os vértices referenciados por `mergeE()` devem existir para que a etapa crie a borda.

Encadear upserts de borda

Assim como acontece com os upserts de vértice, é fácil encadear etapas `mergeE()` para solicitações em lote:

```
g.mergeE([(from): 'v-1', (to): 'v-2', (T.label): 'KNOWS']).
  mergeE([(from): 'v-2', (to): 'v-3', (T.label): 'KNOWS']).
  mergeE([(from): 'v-3', (to): 'v-4', (T.label): 'KNOWS']).
  id()
```

Combinar upserts de vértice e borda

Às vezes, convém aplicar upserts aos vértices e às bordas que os conectam. Você pode misturar os exemplos de lote apresentados aqui. O seguinte exemplo aplica upserts a três vértices e duas bordas:

```
g.mergeV([(id):'v-1']).
    option(onCreate, [(label): 'PERSON', 'email': 'person-1@example.org']).
  mergeV([(id):'v-2']).
    option(onCreate, [(label): 'PERSON', 'email': 'person-2@example.org']).
  mergeV([(id):'v-3']).
    option(onCreate, [(label): 'PERSON', 'email': 'person-3@example.org']).
  mergeE([(from): 'v-1', (to): 'v-2', (T.label): 'KNOWS']).
  mergeE([(from): 'v-2', (to): 'v-3', (T.label): 'KNOWS']).
 id()
```

Misturar upserts e inserções

Às vezes, convém aplicar upserts aos vértices e às bordas que os conectam. Você pode misturar os exemplos de lote apresentados aqui. O seguinte exemplo aplica upserts a três vértices e duas bordas:

Os upserts normalmente avançam um elemento por vez. Se você seguir os padrões de upsert apresentados aqui, cada operação de upsert emitirá um único percurso, o que faz com que a operação subsequente seja executada apenas uma vez.

No entanto, às vezes convém misturar upserts com inserções. Esse pode ser o caso, por exemplo, caso você use bordas para representar instâncias de ações ou eventos. Uma solicitação pode usar upserts para garantir que todos os vértices necessários existam e, depois, usar inserções para adicionar bordas. Com solicitações desse tipo, preste atenção ao número potencial de percursos emitidos por cada operação.

Considere o seguinte exemplo, que mistura upserts e inserções para adicionar bordas que representam eventos no grafo:

```
// Fully optimized, but inserts too many edges
g.mergeV([(id):'v-1']).
    option(onCreate, [(label): 'PERSON', 'email': 'person-1@example.org']).
  mergeV([(id):'v-2']).
    option(onCreate, [(label): 'PERSON', 'email': 'person-2@example.org']).
  mergeV([(id):'v-3']).
    option(onCreate, [(label): 'PERSON', 'email': 'person-3@example.org']).
  mergeV([(T.id): 'c-1', (T.label): 'CITY', name: 'city-1']).
  V('p-1', 'p-2').
  addE('FOLLOWED').to(V('p-1')).
  V('p-1', 'p-2', 'p-3').
  addE('VISITED').to(V('c-1')).
  id()
```

A consulta deve inserir cinco bordas: duas bordas SEGUIDAS e três bordas VISITADAS. No entanto, a consulta, conforme escrita, insere oito bordas: duas SEGUIDAS e seis VISITADAS. A razão disso é que a operação que insere as duas bordas SEGUIDAS emite dois percursos, fazendo com que a operação de inserção subsequente, que insere três bordas, seja executada duas vezes.

A correção é adicionar uma etapa `fold()` após cada operação que possa emitir mais de um percurso:

```
g.mergeV([(T.id): 'v-1', (T.label): 'PERSON', email: 'person-1@example.org']).
  mergeV([(T.id): 'v-2', (T.label): 'PERSON', email: 'person-2@example.org']).
  mergeV([(T.id): 'v-3', (T.label): 'PERSON', email: 'person-3@example.org']).
  mergeV([(T.id): 'c-1', (T.label): 'CITY', name: 'city-1']).
  V('p-1', 'p-2').
  addE('FOLLOWED').
```

```
    to(V('p-1')).
  fold().
  V('p-1', 'p-2', 'p-3').
  addE('VISITED').
    to(V('c-1')).
  id()
```

Aqui, inserimos uma etapa `fold()` após a operação que insere bordas SEGUIDAS. Isso ocasiona um único percurso, o que faz com que a operação subsequente seja executada apenas uma vez.

A desvantagem dessa abordagem é que a consulta agora não está totalmente otimizada, porque `fold()` não está otimizado. A operação de inserção após `fold()` agora também não será otimizada.

Se você precisar usar `fold()` para reduzir o número de percursos em nome das etapas subsequentes, tente ordenar suas operações de forma que as mais baratas ocupem a parte não otimizada da consulta.

Definindo a cardinalidade

A cardinalidade padrão para propriedades de vértice em Netuno está definida, o que significa que, ao usar mergeV (), todos os valores fornecidos no mapa receberão essa cardinalidade. Para usar a cardinalidade única, você deve ser explícito em seu uso. A partir da TinkerPop versão 3.7.0, há uma nova sintaxe que permite que a cardinalidade seja fornecida como parte do mapa, conforme mostrado no exemplo a seguir:

```
g.mergeV([(T.id): 1234]).
  option(onMatch, ['age': single(20), 'name': single('alice'), 'city': set('miami')])
```

Como alternativa, você pode definir a cardinalidade como padrão para isso da `option` seguinte maneira:

```
// age and name are set to single cardinality by default
g.mergeV([(T.id): 1234]).
  option(onMatch, ['age': 22, 'name': 'alice', 'city': set('boston')], single)
```

Há menos opções para definir a cardinalidade `mergeV()` antes da versão 3.7.0. A abordagem geral é voltar à `property()` etapa da seguinte forma:

```
g.mergeV([(T.id): '1234']).
```

```
  option(onMatch, sideEffect(property(single,'age', 20).
  property(set,'city','miami')).constant([:]))
```

> **Note**
>
> Essa abordagem só funcionará `mergeV()` quando for usada com uma etapa inicial. Portanto, você não seria capaz de encadear `mergeV()` em uma única travessia, pois a primeira etapa `mergeV()` após a etapa inicial que usa essa sintaxe produzirá um erro caso o percurso de entrada seja um elemento gráfico. Nesse caso, você gostaria de dividir suas `mergeV()` chamadas em várias solicitações, onde cada uma pode ser uma etapa inicial.

Criar upserts eficientes do Gremlin com `fold()/coalesce()/unfold()`

Um upsert (ou inserção condicional) reutilizará um vértice ou uma borda se já existir ou criará um desses elementos se não existir. Upserts eficientes podem fazer uma diferença significativa no desempenho das consultas do Gremlin.

Esta página mostra como usar o padrão `fold()/coalesce()/unfold()` do Gremlin para criar upserts eficientes. No entanto, com o lançamento da TinkerPop versão 3.6.x introduzida no Neptune na versão 1.2.1.0 do motor, as etapas novas `mergeV()` e são preferíveis na maioria dos casos. `mergeE()` O padrão `fold()/coalesce()/unfold()` descrito aqui ainda pode ser útil em algumas situações complexas, mas, em geral, use `mergeV()` e `mergeE()` se possível, conforme descrito em Criar surtos eficientes com as etapas `mergeV()` e `mergeE()` do Gremlin..

Os upserts permitem que você escreva operações de inserção idempotentes: não importa quantas vezes você execute essa operação, o resultado geral é o mesmo. Isso é útil em cenários de gravação altamente simultâneos em que modificações simultâneas na mesma parte do grafo podem forçar a reversão de uma ou mais transações com `ConcurrentModificationException`, exigindo uma nova tentativa.

Por exemplo, a consulta a seguir aplica upserts a um vértice procurando primeiro pelo vértice especificado no conjunto de dados e, depois, dobrando os resultados em uma lista. No primeiro percurso fornecido para a etapa `coalesce()`, a consulta então desdobra essa lista. Se a lista desdobrada não estiver vazia, os resultados serão emitidos pelo `coalesce()`. Se, no entanto, `unfold()` gerar uma coleção vazia porque o vértice não existe no momento, `coalesce()` avaliará o segundo percurso com o qual foi fornecida e, nesse segundo percurso, a consulta criará o vértice ausente.

```
g.V('v-1').fold()
          .coalesce(
             unfold(),
             addV('Person').property(id, 'v-1')
                           .property('email', 'person-1@example.org')
           )
```

Usar uma forma otimizada de **`coalesce()`** para upserts

O Neptune pode otimizar a expressão `fold().coalesce(unfold(), ...)` para fazer atualizações de throughput, mas essa otimização só funcionará se as duas partes do `coalesce()` gerarem um vértice ou uma borda, nada mais. Se você tentar gerar algo diferente, como uma propriedade, de qualquer parte do `coalesce()`, a otimização do Neptune não ocorrerá. A consulta poderá ser bem-sucedida, mas não funcionará tão bem quanto uma versão otimizada, principalmente em grandes conjuntos de dados.

Como as consultas de upsert não otimizadas aumentam os tempos de execução e reduzem o throughput, vale a pena usar o endpoint `explain` do Gremlin para determinar se uma consulta de upsert está totalmente otimizada. Ao revisar os planos `explain`, procure linhas que comecem com `+ not converted into Neptune steps` e `WARNING: >>`. Por exemplo: .

```
+ not converted into Neptune steps: [FoldStep, CoalesceStep([[UnfoldStep],
 [AddEdgeSte...
WARNING: >> FoldStep << is not supported natively yet
```

Esses avisos podem ajudar você a identificar as partes de uma consulta que estão impedindo que ela seja totalmente otimizada.

Às vezes, não é possível otimizar totalmente uma consulta. Nessas situações, você deve tentar colocar as etapas que não podem ser otimizadas no final da consulta, permitindo que o mecanismo otimize o máximo de etapas possível. Essa técnica é usada em alguns exemplos de upserts em lote, em que todos os upserts otimizados para um conjunto de vértices ou bordas são executados antes que qualquer modificação adicional, possivelmente não otimizada, seja aplicada aos mesmos vértices ou bordas.

Agrupar upserts em lote para melhorar o throughput

Para cenários de gravação de throughput, é possível encadear as etapas para aplicar upserts a vértices e bordas em lote. O agrupamento em lote reduz a sobrecarga transacional de aplicar upserts

a um grande número de vértices e bordas. Depois, é possível melhorar ainda mais o throughput aplicando upserts às solicitações em lote paralelamente usando vários clientes.

Como regra, recomendamos aplicar upserts a cerca de duzentos registros por solicitação em lote. Registro é um único rótulo ou propriedade de vértice ou borda. Um vértice com um único rótulo e quatro propriedades, por exemplo, cria cinco registros. Uma borda com um rótulo e uma única propriedade cria dois registros. Se você quiser aplicar upserts a lotes de vértices, cada um com um único rótulo e quatro propriedades, deverá começar com um tamanho de lote de quarenta, porque `200 / (1 + 4) = 40`.

É possível experimentar o tamanho do lote. Um valor de duzentos registros por lote é um bom ponto de partida, mas o tamanho ideal pode ser maior ou menor, dependendo da workload. Observe, no entanto, que o Neptune pode limitar o número total de etapas do Gremlin por solicitação. Esse limite não está documentado, mas, por segurança, tente garantir que suas solicitações não contenham mais de 1.500 etapas do Gremlin. O Neptune pode rejeitar grandes solicitações em lote com mais de 1.500 etapas.

Para aumentar o throughput, é possível inverter lotes em paralelo usando vários clientes (consulte Criação de gravações eficientes com multi-thread do Gremlin). O número de clientes deve ser igual ao número de threads de operador em sua instância de gravador do Neptune, que normalmente é duas vezes o número de vCPUs no servidor. Por exemplo, uma instância `r5.8xlarge` tem 32 vCPUs e 64 threads de operador. Para cenários de gravação de throughput usando um `r5.8xlarge`, você usaria 64 clientes gravando upserts em lote no Neptune em paralelo.

Cada cliente deve enviar uma solicitação em lote e aguardar a conclusão da solicitação antes de enviar outra solicitação. Embora os vários clientes funcionem paralelamente, cada cliente individual envia solicitações em série. Isso garante que o servidor receba um fluxo constante de solicitações que ocupem todos os threads de operador sem inundar a fila de solicitações do lado do servidor (consulte Dimensionar instâncias de banco de dados em um cluster de banco de dados do Neptune).

Tentar evitar etapas que gerem vários percursos

Quando uma etapa do Gremlin é executada, ela pega um percurso de entrada e emite um ou mais percursos de saída. O número de percursos emitidos por uma etapa determina o número de vezes que a próxima etapa é executada.

Normalmente, ao realizar operações em lote, é recomendável que cada operação, como o vértice de upsert A, seja executada uma vez, para que a sequência de operações seja a seguinte: vértice de upsert A, depois vértice de upsert B, vértice de upsert C, etc. Desde que uma etapa crie ou

modifique somente um elemento, ela emite somente um percurso, e as etapas que representam a próxima operação serão executadas somente uma vez. Se, por outro lado, uma operação criar ou modificar mais de um elemento, ela emitirá vários percursos o que, por sua vez, faz com que as etapas subsequentes sejam executadas várias vezes, uma vez por percurso emitido. Isso pode fazer com que o banco de dados execute trabalho adicional desnecessário e, em alguns casos, pode ocasionar a criação de vértices, bordas ou valores de propriedades adicionais indesejados.

Um exemplo de como as coisas podem dar errado é uma consulta como `g.V().addV()`. Essa consulta simples adiciona um vértice para cada vértice encontrado no grafo, porque `V()` emite um percurso para cada vértice no grafo e cada um desses percursos aciona uma chamada para `addV()`.

Consulte Misturar upserts e inserções para saber como lidar com operações que podem emitir vários percursos.

Aplicar upserts a vértices

É possível usar um ID de vértice para determinar se existe um vértice correspondente. Essa é a abordagem preferencial, porque o Neptune otimiza upserts para casos de uso altamente simultâneos em torno de IDs. Por exemplo, a seguinte consulta criará um vértice com um ID específico, se ele ainda não existir, ou o reutilizará se existir:

```
g.V('v-1')
 .fold()
  .coalesce(unfold(),
            addV('Person').property(id, 'v-1')
                          .property('email', 'person-1@example.org'))
  .id()
```

Observe que essa consulta termina com uma etapa `id()`. Embora não seja estritamente necessário para aplicar upserts ao vértice, adicionar uma etapa `id()` ao final de uma consulta de upsert garante que o servidor não serialize todas as propriedades do vértice de volta para o cliente, o que ajuda a reduzir o custo de bloqueio da consulta.

Você também pode usar uma propriedade de vértice para determinar se o vértice existe:

```
g.V()
 .hasLabel('Person')
 .has('email', 'person-1@example.org')
 .fold()
```

```
 .coalesce(unfold(),
           addV('Person').property('email', 'person-1@example.org'))
 .id()
```

Se possível, use seus próprios IDs fornecidos pelo usuário para criar vértices e use esses IDs para determinar se existe um vértice durante uma operação de upsert. Isso permite que o Neptune otimize os upserts ao redor dos IDs. Um upsert baseado em ID pode ser significativamente mais eficiente do que um upsert baseado em propriedade em cenários de modificações altamente simultâneas.

Encadear upserts de vértice

É possível encadear upserts de vértice para inseri-los em um lote:

```
g.V('v-1')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-1')
                         .property('email', 'person-1@example.org'))
 .V('v-2')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-2')
                         .property('email', 'person-2@example.org'))
 .V('v-3')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-3')
                         .property('email', 'person-3@example.org'))
 .id()
```

Aplicar upserts a bordas

É possível usar IDs de borda para aplicar upserts a bordas da mesma forma que a vértices usando IDs de vértice personalizados. Novamente, essa é a abordagem preferencial porque permite que o Neptune otimize a consulta. Por exemplo, a consulta a seguir cria uma borda com base em seu ID de borda, se ela ainda não existir, ou a reutiliza se existir. A consulta também usará os IDs dos vértices `from` e `to` se precisar criar uma borda.

```
g.E('e-1')
 .fold()
 .coalesce(unfold(),
```

```
           addE('KNOWS').from(V('v-1'))
                        .to(V('v-2'))
                        .property(id, 'e-1'))
 .id()
```

Muitas aplicações usam IDs de vértice personalizados, mas deixam o Neptune gerar IDs de borda. Se você não sabe o ID de uma borda, mas sim os IDs de vértice `from` e `to`, pode usar esta formulação para aplicar upsert a uma borda:

```
g.V('v-1')
 .outE('KNOWS')
 .where(inV().hasId('v-2'))
 .fold()
 .coalesce(unfold(),
           addE('KNOWS').from(V('v-1'))
                        .to(V('v-2')))
 .id()
```

Observe que a etapa de vértice na cláusula `where()` deve ser `inV()` (ou `outV()` se você já usou `inE()` para encontrar a borda), não `otherV()`. Não use `otherV()` aqui, ou a consulta não será otimizada e o desempenho será prejudicado. Por exemplo, o Neptune não otimizaria a seguinte consulta:

```
// Unoptimized upsert, because of otherV()
g.V('v-1')
 .outE('KNOWS')
 .where(otherV().hasId('v-2'))
 .fold()
 .coalesce(unfold(),
           addE('KNOWS').from(V('v-1'))
                        .to(V('v-2')))
 .id()
```

Se você não souber os IDs de borda ou vértice de antemão, você poderá aplicar o upsert usando as propriedades do vértice:

```
g.V()
 .hasLabel('Person')
 .has('name', 'person-1')
 .outE('LIVES_IN')
 .where(inV().hasLabel('City').has('name', 'city-1'))
```

```
  .fold()
  .coalesce(unfold(),
            addE('LIVES_IN').from(V().hasLabel('Person')
                                    .has('name', 'person-1'))
                            .to(V().hasLabel('City')
                                   .has('name', 'city-1')))
  .id()
```

Assim como acontece com os upserts de vértice, é preferível usar upserts de borda baseados em ID usando um ID de borda ou os IDS de vértice `from` e `to`, em vez de upserts baseados em propriedades, para que Neptune possa otimizar totalmente o upsert.

Conferir a existência dos vértices **`from`** e **`to`**

Observe a construção das etapas que criam uma borda: `addE().from().to()`. Essa construção garante que a consulta confira a existência dos vértices `from` e `to`. Se alguma delas não existir, a consulta vai gerar um erro da seguinte forma:

```
{
  "detailedMessage": "Encountered a traverser that does not map to a value for child...
  "code": "IllegalArgumentException",
  "requestId": "..."
}
```

Se for possível que os vértices `from` ou `to` não existam, você deverá tentar aplicar upsert a eles antes de aplicar upsert à borda entre eles. Consulte Combinar upserts de vértice e borda.

Existe uma construção alternativa para criar uma borda que você não deve usar: `V().addE().to()`. Ela só adicionará uma borda se o vértice `from` existir. Se o vértice `to` não existir, a consulta gerará um erro, conforme descrito anteriormente, mas se o vértice `from` não existir, ele falhará silenciosamente ao inserir uma borda, sem gerar nenhum erro. Por exemplo, o seguinte upsert será concluído sem aplicar upsert a uma borda se o vértice `from` não existir:

```
// Will not insert edge if from vertex does not exist
g.V('v-1')
  .outE('KNOWS')
  .where(inV().hasId('v-2'))
  .fold()
  .coalesce(unfold(),
            V('v-1').addE('KNOWS')
                    .to(V('v-2')))
```

```
  .id()
```

Encadear upserts de borda

Se você quiser encadear upserts de borda para criar uma solicitação em lote, deverá começar cada upsert com uma pesquisa de vértice, mesmo que já conheça os IDs de borda.

Se você já sabe os IDs das bordas às quais deseja aplicar upserts e os IDs dos vértices `from` e `to`, poderá usar esta formulação:

```
g.V('v-1')
 .outE('KNOWS')
 .hasId('e-1')
 .fold()
 .coalesce(unfold(),
           V('v-1').addE('KNOWS')
                   .to(V('v-2'))
                   .property(id, 'e-1'))
 .V('v-3')
 .outE('KNOWS')
 .hasId('e-2').fold()
 .coalesce(unfold(),
           V('v-3').addE('KNOWS')
                   .to(V('v-4'))
                   .property(id, 'e-2'))
 .V('v-5')
 .outE('KNOWS')
 .hasId('e-3')
 .fold()
 .coalesce(unfold(),
           V('v-5').addE('KNOWS')
                   .to(V('v-6'))
                   .property(id, 'e-3'))
 .id()
```

Talvez o cenário mais comum de upsert de bordas em lote seja você saber os IDs dos vértices `from` e `to`, mas não os IDs das bordas às quais deseja aplicar upserts. Nesse caso, use a seguinte formulação:

```
g.V('v-1')
 .outE('KNOWS')
 .where(inV().hasId('v-2'))
```

```
  .fold()
  .coalesce(unfold(),
            V('v-1').addE('KNOWS')
                    .to(V('v-2')))

  .V('v-3')
  .outE('KNOWS')
  .where(inV().hasId('v-4'))
  .fold()
  .coalesce(unfold(),
            V('v-3').addE('KNOWS')
                    .to(V('v-4')))
  .V('v-5')
  .outE('KNOWS')
  .where(inV().hasId('v-6'))
  .fold()
  .coalesce(unfold(),
            V('v-5').addE('KNOWS').to(V('v-6')))
  .id()
```

Se você já sabe os IDs das bordas às quais deseja aplicar upserts, mas não sabe os IDs dos vértices `from` e `to` (situação incomum), poderá usar esta formulação:

```
g.V()
  .hasLabel('Person')
  .has('email', 'person-1@example.org')
  .outE('KNOWS')
  .hasId('e-1')
  .fold()
  .coalesce(unfold(),
            V().hasLabel('Person')
               .has('email', 'person-1@example.org')
               .addE('KNOWS')
               .to(V().hasLabel('Person')
                      .has('email', 'person-2@example.org'))
                      .property(id, 'e-1'))
  .V()
  .hasLabel('Person')
  .has('email', 'person-3@example.org')
  .outE('KNOWS')
  .hasId('e-2')
  .fold()
  .coalesce(unfold(),
```

```
          V().hasLabel('Person')
             .has('email', 'person-3@example.org')
             .addE('KNOWS')
             .to(V().hasLabel('Person')
                    .has('email', 'person-4@example.org'))
             .property(id, 'e-2'))
 .V()
 .hasLabel('Person')
 .has('email', 'person-5@example.org')
 .outE('KNOWS')
 .hasId('e-1')
 .fold()
 .coalesce(unfold(),
          V().hasLabel('Person')
             .has('email', 'person-5@example.org')
             .addE('KNOWS')
             .to(V().hasLabel('Person')
                    .has('email', 'person-6@example.org'))
                    .property(id, 'e-3'))
 .id()
```

Combinar upserts de vértice e borda

Às vezes, convém aplicar upserts aos vértices e às bordas que os conectam. Você pode misturar os exemplos de lote apresentados aqui. O seguinte exemplo aplica upserts a três vértices e duas bordas:

```
g.V('p-1')
 .fold()
 .coalesce(unfold(),
          addV('Person').property(id, 'p-1')
                        .property('email', 'person-1@example.org'))
 .V('p-2')
 .fold()
 .coalesce(unfold(),
          addV('Person').property(id, 'p-2')
                        .property('name', 'person-2@example.org'))
 .V('c-1')
 .fold()
 .coalesce(unfold(),
          addV('City').property(id, 'c-1')
                      .property('name', 'city-1'))
 .V('p-1')
```

```
 .outE('LIVES_IN')
 .where(inV().hasId('c-1'))
 .fold()
 .coalesce(unfold(),
           V('p-1').addE('LIVES_IN')
                   .to(V('c-1')))
 .V('p-2')
 .outE('LIVES_IN')
 .where(inV().hasId('c-1'))
 .fold()
 .coalesce(unfold(),
           V('p-2').addE('LIVES_IN')
                   .to(V('c-1')))
 .id()
```

Misturar upserts e inserções

Às vezes, convém aplicar upserts aos vértices e às bordas que os conectam. Você pode misturar os exemplos de lote apresentados aqui. O seguinte exemplo aplica upserts a três vértices e duas bordas:

Os upserts normalmente avançam um elemento por vez. Se você seguir os padrões de upsert apresentados aqui, cada operação de upsert emitirá um único percurso, o que faz com que a operação subsequente seja executada apenas uma vez.

No entanto, às vezes convém misturar upserts com inserções. Esse pode ser o caso, por exemplo, caso você use bordas para representar instâncias de ações ou eventos. Uma solicitação pode usar upserts para garantir que todos os vértices necessários existam e, depois, usar inserções para adicionar bordas. Com solicitações desse tipo, preste atenção ao número potencial de percursos emitidos por cada operação.

Considere o seguinte exemplo, que mistura upserts e inserções para adicionar bordas que representam eventos no grafo:

```
// Fully optimized, but inserts too many edges
g.V('p-1')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'p-1')
                         .property('email', 'person-1@example.org'))
```

```
 .V('p-2')
 .fold()
 .coalesce(unfold(),
          addV('Person').property(id, 'p-2')
                        .property('name', 'person-2@example.org'))
 .V('p-3')
 .fold()
 .coalesce(unfold(),
          addV('Person').property(id, 'p-3')
                        .property('name', 'person-3@example.org'))
 .V('c-1')
 .fold()
 .coalesce(unfold(),
          addV('City').property(id, 'c-1')
                      .property('name', 'city-1'))
 .V('p-1', 'p-2')
 .addE('FOLLOWED')
 .to(V('p-1'))
 .V('p-1', 'p-2', 'p-3')
 .addE('VISITED')
 .to(V('c-1'))
 .id()
```

A consulta deve inserir cinco bordas: duas bordas SEGUIDAS e três bordas VISITADAS. No entanto, a consulta, conforme escrita, insere oito bordas: duas SEGUIDAS e seis VISITADAS. A razão disso é que a operação que insere as duas bordas SEGUIDAS emite dois percursos, fazendo com que a operação de inserção subsequente, que insere três bordas, seja executada duas vezes.

A correção é adicionar uma etapa `fold()` após cada operação que possa emitir mais de um percurso:

```
g.V('p-1')
 .fold()
 .coalesce(unfold(),
          addV('Person').property(id, 'p-1')
                        .property('email', 'person-1@example.org'))
 .V('p-2')
 .fold()
 .coalesce(unfold(),
          addV('Person').property(id, 'p-2').
                        .property('name', 'person-2@example.org'))
 .V('p-3')
 .fold()
```

```
.coalesce(unfold(),
          addV('Person').property(id, 'p-3').
                        .property('name', 'person-3@example.org'))
.V('c-1')
.fold().
.coalesce(unfold(),
           addV('City').property(id, 'c-1').
                       .property('name', 'city-1'))
.V('p-1', 'p-2')
.addE('FOLLOWED')
.to(V('p-1'))
.fold()
.V('p-1', 'p-2', 'p-3')
.addE('VISITED')
.to(V('c-1')).
.id()
```

Aqui, inserimos uma etapa `fold()` após a operação que insere bordas SEGUIDAS. Isso ocasiona um único percurso, o que faz com que a operação subsequente seja executada apenas uma vez.

A desvantagem dessa abordagem é que a consulta agora não está totalmente otimizada, porque `fold()` não está otimizado. A operação de inserção após `fold()` agora não será otimizada.

Se você precisar usar `fold()` para reduzir o número de percursos em nome das etapas subsequentes, tente ordenar suas operações de forma que as mais baratas ocupem a parte não otimizada da consulta.

Upserts que modificam vértices e bordas existentes

Às vezes, você deseja criar um vértice ou uma borda, se não existirem, e depois adicionar ou atualizar uma propriedade, independentemente de ser uma borda ou um vértice novo ou existente.

Para adicionar ou modificar uma propriedade, use a etapa `property()`. Use essa etapa fora da etapa `coalesce()`. Se você tentar modificar a propriedade de um vértice ou uma borda existente dentro da etapa `coalesce()`, a consulta poderá não ser otimizada pelo mecanismo de consulta do Neptune.

A consulta a seguir adiciona ou atualiza uma propriedade do contador em cada vértice com upserts aplicados. Cada etapa `property()` tem uma cardinalidade única para garantir que os novos valores substituam todos os valores existentes, em vez de serem adicionados a um conjunto de valores existentes.

```
g.V('v-1')
 .fold()
 .coalesce(unfold(),
          addV('Person').property(id, 'v-1')
                        .property('email', 'person-1@example.org'))
 .property(single, 'counter', 1)
 .V('v-2')
 .fold()
 .coalesce(unfold(),
          addV('Person').property(id, 'v-2')
                        .property('email', 'person-2@example.org'))
 .property(single, 'counter', 2)
 .V('v-3')
 .fold()
 .coalesce(unfold(),
          addV('Person').property(id, 'v-3')
                        .property('email', 'person-3@example.org'))
 .property(single, 'counter', 3)
 .id()
```

Se você tiver um valor de propriedade, como um valor de carimbo de data e hora `lastUpdated`, que se aplique a todos os elementos alterados, poderá adicioná-lo ou atualizá-lo no final da consulta:

```
g.V('v-1')
 .fold()
 .coalesce(unfold(),
          addV('Person').property(id, 'v-1')
                        .property('email', 'person-1@example.org'))
 .V('v-2').
 .fold().
 .coalesce(unfold(),
          addV('Person').property(id, 'v-2')
                        .property('email', 'person-2@example.org'))
 .V('v-3')
 .fold()
 .coalesce(unfold(),
          addV('Person').property(id, 'v-3')
                        .property('email', 'person-3@example.org'))
 .V('v-1', 'v-2', 'v-3')
 .property(single, 'lastUpdated', datetime('2020-02-08'))
 .id()
```

Se houver condições adicionais que determinem se uma borda ou um vértice deve ou não ser modificado posteriormente, você poderá usar uma etapa `has()` para filtrar os elementos aos quais uma modificação será aplicada. O exemplo a seguir usa uma etapa `has()` para filtrar vértices com upserts aplicados com base no valor da propriedade `version`. Depois, a consulta é atualizada para a `version` 3 de qualquer vértice cuja `version` seja inferior a 3:

```
g.V('v-1')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-1')
                         .property('email', 'person-1@example.org')
                         .property('version', 3))
 .V('v-2')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-2')
                         .property('email', 'person-2@example.org')
                         .property('version', 3))
 .V('v-3')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-3')
                         .property('email', 'person-3@example.org')
                         .property('version', 3))
 .V('v-1', 'v-2', 'v-3')
 .has('version', lt(3))
 .property(single, 'version', 3)
 .id()
```

Analisar a execução de consulta do Neptune usando o **`explain`** do Gremlin

O Amazon Neptune adicionou um atributo do Gremlin denominado explain. Esse atributo é uma ferramenta de autoatendimento para compreender a abordagem da execução realizada pelo mecanismo do Neptune. Você o invoca adicionando um parâmetro `explain` a uma chamada HTTP que envia uma consulta do Gremlin.

O recurso `explain` fornece informações sobre a estrutura lógica dos planos de execução da consulta. Você pode usar essas informações para identificar possíveis gargalos de execução e de

avaliação e ajustar a consulta, conforme explicado em Ajustar consultas do Gremlin. Você também pode usar as dicas de consulta para melhorar os planos de execução das consultas.

> **Note**
>
> Esse atributo está disponível a partir da Versão 1.0.1.0.200463.0 (15/10/2019).

Tópicos

- Noções básicas de como as consultas do Gremlin funcionam no Neptune
- Usar a API explain do Gremlin no Neptune
- API profile do Gremlin no Neptune
- Ajustar consultas do Gremlin usando explain e profile
- Suporte nativo para etapas do Gremlin no Amazon Neptune

Noções básicas de como as consultas do Gremlin funcionam no Neptune

Para aproveitar ao máximo o recurso os relatórios `explain` e `profile` do Gremlin no Amazon Neptune, é útil compreender algumas informações básicas sobre consultas do Gremlin.

Tópicos

- Declarações do Gremlin no Neptune
- Como o Neptune processa consultas do Gremlin usando índices de declaração
- Como as consultas do Gremlin são processadas no Neptune

Declarações do Gremlin no Neptune

Os dados do grafo de propriedades no Amazon Neptune são compostos por declarações de quatro posições (quadrante). Cada uma dessas instruções representa uma unidade atômica individual de dados do gráfico de propriedades. Para ter mais informações, consulte Modelo de dados de grafo do Neptune. Semelhante ao modelo de dados Resource Description Framework (RDF — Estrutura de descrição de recursos), essas quatro posições são as seguintes:

- `subject (S)`
- `predicate (P)`

- `object (O)`
- `graph (G)`

Cada instrução é uma declaração sobre um ou mais recursos. Por exemplo, uma instrução pode confirmar a existência de um relacionamento entre dois recursos ou anexar uma propriedade (par de valor-chave) a algum recurso.

Você pode pensar no predicado como o verbo da instrução, que descreve o tipo de relacionamento ou propriedade. O objeto é o destino do relacionamento ou o valor da propriedade. A posição do gráfico é opcional e pode ser usada de muitas maneiras diferentes. Para os dados do grafo de propriedades (PG) do Neptune, ela não é usada (grafo nulo) ou é usada para representar o identificador de uma borda. Um conjunto de instruções com identificadores de recursos compartilhados cria um gráfico.

Há três classes de declaração no modelo de dados do grafo de propriedades do Neptune:

Tópicos

- Instruções de rótulo de vértice do Gremlin
- Instruções de borda do Gremlin
- Instruções de propriedade do Gremlin

Instruções de rótulo de vértice do Gremlin

As declarações de rótulo de vértice no Neptune têm duas finalidades:

- Elas rastreiam os rótulos de um vértice.
- A presença de pelo menos uma dessas instruções é o que sugere a existência de um determinado vértice no gráfico.

O assunto dessas instruções é um identificador de vértice, e o objeto é um rótulo, ambos especificados pelo usuário. Você usa um predicado fixo especial para essas instruções, exibido como `<~label>`, e um identificador de gráfico padrão (o gráfico nulo), exibido como `<~>`.

Por exemplo, considere a seguinte travessia `addV`:

```
g.addV("Person").property(id, "v1")
```

Essa travessia resulta na adição da seguinte instrução ao gráfico.

```
StatementEvent[Added(<v1> <~label> <Person> <~>) .]
```

Instruções de borda do Gremlin

Uma declaração de borda do Gremlin é o que sugere a existência de uma borda entre dois vértices em um grafo no Neptune. O assunto (S) de uma instrução de borda é o vértice de origem `from`. O predicado (P) é um rótulo de borda fornecido pelo usuário. O objeto (O) é o vértice de destino `to`. O gráfico (G) é um identificador de borda fornecido pelo usuário.

Por exemplo, considere a seguinte travessia `addE`:

```
g.addE("knows").from(V("v1")).to(V("v2")).property(id, "e1")
```

A travessia resulta na adição da seguinte instrução ao gráfico.

```
StatementEvent[Added(<v1> <knows> <v2> <e1>) .]
```

Instruções de propriedade do Gremlin

Uma declaração de propriedade do Gremlin no Neptune declara um valor de propriedade individual para um vértice ou uma borda. O assunto é um identificador de vértice ou de borda fornecido pelo usuário. O predicado é o nome da propriedade (chave), e o objeto é o valor da propriedade individual. O gráfico (G) é novamente o identificador de gráfico padrão, o gráfico nulo, exibido como `<~>`.

Considere o seguinte exemplo.

```
g.V("v1").property("name", "John")
```

Essa instrução resulta no seguinte.

```
StatementEvent[Added(<v1> <name> "John" <~>) .]
```

As instruções de propriedade diferem das outras porque seu objeto é um valor primitivo (`string`, `date`, `byte`, `short`, `int`, `long`, `float` ou `double`). O objeto não é um identificador de recurso que possa ser usado como assunto de outra declaração.

Para várias propriedades, cada valor de propriedade individual no conjunto recebe sua própria instrução.

```
g.V("v1").property(set, "phone", "956-424-2563").property(set, "phone", "956-354-3692
 (tel:9563543692)")
```

Isso resulta no seguinte.

```
StatementEvent[Added(<v1> <phone> "956-424-2563" <~>) .]
StatementEvent[Added(<v1> <phone> "956-354-3692" <~>) .]
```

Como o Neptune processa consultas do Gremlin usando índices de declaração

As declarações são acessadas no Amazon Neptune por meio de três índices de declaração, conforme detalhado em Como as declarações são indexadas no Neptune. O Neptune extrai um padrão de declaração de uma consulta do Gremlin na qual algumas posições são conhecidas e o restante é deixado para descoberta por pesquisa de índice.

O Neptune pressupõe que o tamanho do esquema do grafo de propriedades não seja grande. Isso significa que o número de rótulos de borda e nomes de propriedade distintos é bastante baixo, resultando em um número total baixo de predicados distintos. O Neptune monitora predicados distintos em um índice separado. Ele usa esse cache de predicados para fazer uma varredura de união de `{ all P x POGS }`, em vez de usar um índice OSGP. Evitar a necessidade de um índice OSGP de travessia reversa economiza espaço de armazenamento e taxa de transferência de carga.

A API Explain/Profile do Gremlin no Neptune permite que você obtenha a contagem de predicados no grafo. Depois, é possível determinar se a aplicação invalida a pressuposição do Neptune de que o esquema do grafo de propriedades é pequeno.

Os exemplos a seguir ajudam a ilustrar como o Neptune usa índices para processar consultas do Gremlin.

Pergunta: quais são os rótulos do vértice **v1**?

```
  Gremlin code:      g.V('v1').label()
  Pattern:           (<v1>, <~label>, ?, ?)
  Known positions:   SP
  Lookup positions:  OG
  Index:             SPOG
```

```
Key range:          <v1>:<~label>:*
```

Pergunta: quais são as bordas externas do vértice **v1** conhecidas?

```
Gremlin code:       g.V('v1').out('knows')
Pattern:            (<v1>, <knows>, ?, ?)
Known positions:    SP
Lookup positions:   OG
Index:              SPOG
Key range:          <v1>:<knows>:*
```

Pergunta: quais vértices têm um rótulo de vértice **Person**?

```
Gremlin code:       g.V().hasLabel('Person')
Pattern:            (?, <~label>, <Person>, <~>)
Known positions:    POG
Lookup positions:   S
Index:              POGS
Key range:          <~label>:<Person>:<~>:*
```

Pergunta: quais são os vértices de/para de uma determinada borda **e1**?

```
Gremlin code:       g.E('e1').bothV()
Pattern:            (?, ?, ?, <e1>)
Known positions:    G
Lookup positions:   SPO
Index:              GPSO
Key range:          <e1>:*
```

Um índice de declaração que o Neptune não tem é um índice OSGP de percurso reverso. Esse índice pode ser usado para reunir todas as bordas de entrada em todos os rótulos de borda, como no exemplo a seguir.

Pergunta: Quais são os vértices adjacentes de entrada **v1**?

```
Gremlin code:       g.V('v1').in()
Pattern:            (?, ?, <v1>, ?)
Known positions:    O
Lookup positions:   SPG
Index:              OSGP  // <-- Index does not exist
```

Como as consultas do Gremlin são processadas no Neptune

No Amazon Neptune, percursos mais complexos podem ser representados por uma série de padrões que criam uma relação com base na definição de variáveis nomeadas que podem ser compartilhadas entre padrões para criar junções. Isso é mostrado no exemplo a seguir.

Pergunta: O que é a área de dois saltos do vértice **v1**?

```
Gremlin code:      g.V('v1').out('knows').out('knows').path()
Pattern:           (?1=<v1>, <knows>, ?2, ?) X Pattern(?2, <knows>, ?3, ?)

The pattern produces a three-column relation (?1, ?2, ?3) like this:
                   ?1     ?2     ?3
                   ================
                   v1     v2     v3
                   v1     v2     v4
                   v1     v5     v6
```

Ao compartilhar a variável ?2 entre os dois padrões (na posição O no primeiro padrão e na posição S do segundo padrão), você cria uma junção dos primeiros vizinhos de salto aos segundos vizinhos de salto. Cada solução Neptune tem vinculações para as três variáveis nomeadas, que podem ser usadas para recriar TinkerPopum Traverser (incluindo informações de caminho).

A primeira etapa no processamento de consultas do Gremlin é analisar a consulta em um objeto TinkerPop Traversal, composto por uma série de etapas. TinkerPop Essas etapas, que fazem parte do TinkerPop projeto Apache de código aberto, são os operadores lógicos e físicos que compõem uma travessia do Gremlin na implementação de referência. Elas são usadas para representar o modelo da consulta. São operadores executáveis que podem produzir soluções de acordo com a semântica do operador que representam. Por exemplo, `.V()` é representado e executado pelo TinkerPop GraphStep.

Como essas off-the-shelf TinkerPop etapas são executáveis, esse TinkerPop Traversal pode executar qualquer consulta do Gremlin e produzir a resposta correta. No entanto, quando executadas em um gráfico grande, TinkerPop as etapas às vezes podem ser muito ineficientes e lentas. Em vez de usá-las, o Neptune tenta converter o percurso em um formulário declarativo composto de grupos de padrões, conforme descrito anteriormente.

No momento, o Neptune não é compatível com todos os operadores (etapas) do Gremlin no mecanismo de consulta nativo. Portanto, ele tenta recolher o maior número possível de etapas em

uma única `NeptuneGraphQueryStep`, que contém o plano de consulta lógica declarativa de todas as etapas que foram convertidas. De maneira ideal, todas as etapas são convertidas. Mas quando é encontrada uma etapa que não pode ser convertida, o Neptune sai da execução nativa e adia toda a execução de consultas desse ponto em diante para as etapas. TinkerPop Ele não tenta entrar e sair da execução nativa.

Depois que as etapas são convertidas em um plano de consulta lógica, o Neptune executa uma série de otimizadores de consulta que reescrevem o plano de consulta com base na análise estática e nas cardinalidades estimadas. Esses otimizadores executam coisas como reordenar operadores com base em contagens de intervalo, remover operadores desnecessários ou redundantes, reorganizar filtros, enviar operadores para grupos diferentes e assim por diante.

Depois que um plano de consulta otimizado é produzido, o Neptune cria um pipeline de operadores físicos que executam o trabalho de execução da consulta. Isso inclui a leitura de dados dos índices de instrução, a execução de junções de vários tipos, a filtragem, a ordenação e assim por diante. O pipeline produz um fluxo de solução que é então convertido novamente em um fluxo de objetos TinkerPop Traverser.

Serialização dos resultados da consulta

Atualmente, o Amazon Neptune depende dos serializadores de mensagens de resposta para converter TinkerPop os resultados da consulta TinkerPop (Traversers) em dados serializados a serem enviados pela rede de volta ao cliente. Esses formatos de serialização tendem a ser bastante detalhados.

Por exemplo, para serializar o resultado de uma consulta de vértice, como `g.V().limit(1)`, o mecanismo de consulta do Neptune deve executar uma única pesquisa para produzir o resultado da consulta. No entanto, o serializador `GraphSON` executaria um grande número de pesquisas adicionais para empacotar o vértice no formato de serialização. Ele teria que executar uma pesquisa para obter o rótulo, uma pesquisa para obter as chaves de propriedades e uma pesquisa por chave de propriedade para o vértice obter todos os valores de cada chave.

Alguns dos formatos de serialização são mais eficientes, mas todos exigem pesquisas adicionais. Além disso, os TinkerPop serializadores não tentam evitar pesquisas duplicadas, muitas vezes resultando na repetição desnecessária de muitas pesquisas.

Isso faz com que seja muito importante escrever suas consultas para que solicitem especificamente apenas as informações de que precisam. Por exemplo, o `g.V().limit(1).id()` retornaria apenas o ID do vértice e eliminaria todas as pesquisas adicionais do serializador. O API `profile`

do Gremlin no Neptune permite que você veja quantas chamadas de pesquisa são feitas durante a execução da consulta e durante a serialização.

Usar a API **`explain`** do Gremlin no Neptune

A API `explain` do Gremlin no Amazon Neptune exibe o plano de consulta que seria executado se uma consulta especificada fosse executada. Como a API não executa realmente a consulta, o plano é retornado quase instantaneamente.

Ela difere da etapa TinkerPop .explain () para poder relatar informações específicas do mecanismo Neptune.

Informações contidas em um relatório **`explain`** do Gremlin

Um relatório de `explain` contém as seguintes informações:

- A string de consulta conforme solicitado.
- O percurso original. Esse é o objeto TinkerPop Traversal produzido pela análise da string de consulta em etapas. TinkerPop É equivalente à consulta original produzida pela execução da consulta `.explain()` em relação ao TinkerPop TinkerGraph.
- O percurso convertido. Essa é a Travessia de Netuno produzida pela conversão da Traversal na representação do plano de consulta lógica de TinkerPop Netuno. Em muitos casos, a TinkerPop travessia inteira é convertida em duas etapas do Neptune: uma que executa a consulta inteira () `NeptuneGraphQueryStep` e outra que converte a saída do mecanismo de consulta Neptune novamente em Traversers (). TinkerPop `NeptuneTraverserConverterStep`
- O percurso otimizado. Essa é a versão otimizada do plano de consulta do Neptune depois que ele é executado por meio de uma série de otimizadores de redução de trabalho estático que reescrevem a consulta com base na análise estática e nas cardinalidades estimadas. Esses otimizadores executam coisas como reordenar operadores com base em contagens de intervalo, remover operadores desnecessários ou redundantes, reorganizar filtros, enviar operadores para grupos diferentes e assim por diante.
- A contagem de predicados. Devido à estratégia de indexação do Neptune descrita anteriormente, ter um grande número de predicados diferentes pode ocasionar problemas de desempenho. Isso é especialmente verdadeiro para consultas que usam operadores de travessia reversa sem rótulo de borda (`.in` ou `.both`). Se esses operadores forem usados e a contagem de predicados for alta o suficiente, o relatório de `explain` exibirá uma mensagem de aviso.
- Informações do DFE. Quando o mecanismo alternativo do DFE está habilitado, os seguintes componentes de percurso podem aparecer no percurso otimizado:

- **`DFEStep`**: uma etapa otimizada do DFE para Neptune no percurso que contém um `DFENode` secundário. `DFEStep` representa a parte do plano de consulta que é executada no mecanismo do DFE.
- **`DFENode`**: contém a representação intermediária como um ou mais `DFEJoinGroupNodes` secundários.
- **`DFEJoinGroupNode`**: representa uma junção de um ou mais elementos `DFENode` ou `DFEJoinGroupNode`.
- **`NeptuneInterleavingStep`**: uma etapa otimizada do DFE para Netuno no percurso que contém uma `DFEStep` secundária.

  Também apresenta um elemento `stepInfo` que contém informações sobre o percurso, como o elemento de fronteira, os elementos do caminho usados, etc. Essas informações são usadas para processar a `DFEStep` secundária.

Uma maneira fácil de descobrir se sua consulta está sendo avaliada pelo DFE é conferir se a saída `explain` contém uma `DFEStep`. Qualquer parte da travessia que não faça parte da não `DFEStep` será executada pelo DFE e será executada pelo motor. TinkerPop

Consulte Exemplo com o DFE habilitado para obter um exemplo de relatório.

Sintaxe de **`explain`** do Gremlin

A sintaxe da API `explain` é a mesma da API HTTP para consulta, com a exceção de que ela usa `/gremlin/explain` como o endpoint, em vez de `/gremlin`, como no exemplo a seguir.

```
curl -X POST https://your-neptune-endpoint:port/gremlin/explain -d
 '{"gremlin":"g.V().limit(1)"}'
```

A consulta anterior produziria a saída a seguir.

```
*******************************************************
               Neptune Gremlin Explain
*******************************************************

Query String
============
g.V().limit(1)

Original Traversal
```

```
==================
[GraphStep(vertex,[]), RangeGlobalStep(0,1)]

Converted Traversal
==================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .]
        }, finishers=[limit(1)], annotations={path=[Vertex(?1):GraphStep], maxVarId=3}
    },
    NeptuneTraverserConverterStep
]

Optimized Traversal
==================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .],
 {estimatedCardinality=INFINITY}
        }, finishers=[limit(1)], annotations={path=[Vertex(?1):GraphStep], maxVarId=3}
    },
    NeptuneTraverserConverterStep
]

Predicates
==========
# of predicates: 18
```

Etapas não convertidas TinkerPop

Idealmente, todas as TinkerPop etapas de uma travessia têm cobertura nativa do operador Neptune. Quando esse não é o caso, o Neptune recorre à execução de etapas devido a lacunas TinkerPop na cobertura do operador. Se um percurso usar uma etapa para a qual o Neptune ainda não tem cobertura nativa, o relatório do `explain` exibirá um aviso mostrando onde ocorreu o déficit.

Quando uma etapa sem um operador de Netuno nativo correspondente é encontrada, toda a travessia daquele ponto em diante é executada TinkerPop usando etapas, mesmo que as etapas subsequentes tenham operadores nativos de Netuno.

A exceção ocorre ao invocar a pesquisa de texto completo do Neptune. O NeptuneSearchStep implementa etapas sem equivalentes nativos como etapas de pesquisa de texto completo.

Exemplo de saída de **`explain`** em que todas as etapas de uma consulta têm equivalentes nativos

Veja a seguir um exemplo de relatório `explain` de uma consulta em que todas as etapas têm equivalentes nativos:

```
*******************************************************
                Neptune Gremlin Explain
*******************************************************

Query String
============
g.V().out()

Original Traversal
==================
[GraphStep(vertex,[]), VertexStep(OUT,vertex)]

Converted Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .]
            PatternNode[(?1, ?5, ?3, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .]
            PatternNode[(?3, <~label>, ?4, <~>) . project ask .]
        }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep], maxVarId=7}
    },
    NeptuneTraverserConverterStep
]

Optimized Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, ?5, ?3, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .],
 {estimatedCardinality=INFINITY}
        }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep], maxVarId=7}
```

```
    },
    NeptuneTraverserConverterStep
]

Predicates
==========
# of predicates: 18
```

Exemplo em que algumas etapas de uma consulta não têm equivalentes nativos

O Neptune lida com `GraphStep` e `VertexStep` de forma nativa, mas se você inserir uma `FoldStep` e uma `UnfoldStep`, a saída de `explain` resultante será diferente:

```
*******************************************************
                Neptune Gremlin Explain
*******************************************************

Query String
============
g.V().fold().unfold().out()

Original Traversal
==================
[GraphStep(vertex,[]), FoldStep, UnfoldStep, VertexStep(OUT,vertex)]

Converted Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .]
        }, annotations={path=[Vertex(?1):GraphStep], maxVarId=3}
    },
    NeptuneTraverserConverterStep
]
+ not converted into Neptune steps: [FoldStep, UnfoldStep, VertexStep(OUT,vertex)]

Optimized Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
```

```
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .],
 {estimatedCardinality=INFINITY}
        }, annotations={path=[Vertex(?1):GraphStep], maxVarId=3}
    },
    NeptuneTraverserConverterStep,
    NeptuneMemoryTrackerStep
]
+ not converted into Neptune steps: [FoldStep, UnfoldStep, VertexStep(OUT,vertex)]

WARNING: >> FoldStep << is not supported natively yet
```

Nesse caso, a `FoldStep` interrompe a execução nativa. Mas até mesmo a `VertexStep` subsequente não é mais tratada nativamente porque ela aparece downstream das etapas de `Fold`/`Unfold`.

Para obter desempenho e economia de custos, é importante que você tente formular percursos para que a quantidade máxima de trabalho possível seja feita de forma nativa dentro do mecanismo de consulta Neptune, em vez de nas implementações por etapas. TinkerPop

Exemplo de uma consulta que usa Neptune full-text-search

A seguinte consulta usa a pesquisa de texto completo do Neptune:

```
g.withSideEffect("Neptune#fts.endpoint", "some_endpoint")
  .V()
  .tail(100)
  .has("Neptune#fts mark*")
  -------
  .has("name", "Neptune#fts mark*")
  .has("Person", "name", "Neptune#fts mark*")
```

A parte `.has("name", "Neptune#fts mark*")` limita a pesquisa a vértices com `name`, enquanto `.has("Person", "name", "Neptune#fts mark*")` limita a pesquisa a vértices com `name` e o rótulo `Person`. Isso resulta no seguinte percurso no relatório `explain`:

```
Final Traversal
[NeptuneGraphQueryStep(Vertex) {
    JoinGroupNode {
        PatternNode[(?1, termid(1,URI), ?2, termid(0,URI)) . project distinct ?1 .],
 {estimatedCardinality=INFINITY}
```

```
    }, annotations={path=[Vertex(?1):GraphStep], maxVarId=4}
}, NeptuneTraverserConverterStep, NeptuneTailGlobalStep(10),
 NeptuneTinkerpopTraverserConverterStep, NeptuneSearchStep {
    JoinGroupNode {
        SearchNode[(idVar=?3, query=mark*, field=name) . project ask .],
 {endpoint=some_endpoint}
    }
    JoinGroupNode {
        SearchNode[(idVar=?3, query=mark*, field=name) . project ask .],
 {endpoint=some_endpoint}
    }
}]
```

Exemplo de uso de **`explain`** quando o DFE está habilitado

Veja um exemplo de relatório `explain` quando o mecanismo de consulta alternativo do DFE está habilitado:

```
********************************************************
                Neptune Gremlin Explain
********************************************************

Query String
============

g.V().as("a").out().has("name", "josh").out().in().where(eq("a"))


Original Traversal
==================
[GraphStep(vertex,[])@[a], VertexStep(OUT,vertex), HasStep([name.eq(josh)]),
 VertexStep(OUT,vertex), VertexStep(IN,vertex), WherePredicateStep(eq(a))]

Converted Traversal
===================
Neptune steps:
[
    DFEStep(Vertex) {
      DFENode {
        DFEJoinGroupNode[ children={
          DFEPatternNode[(?1, <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>, ?2,
 <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph>) . project DISTINCT[?1]
 {rangeCountEstimate=unknown}],
```

```
          DFEPatternNode[(?1, ?3, ?4, ?5) . project ALL[?1, ?4] graphFilters=(!
= <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph> . ),
 {rangeCountEstimate=unknown}]
        }, {rangeCountEstimate=unknown}
        ]
      } [Vertex(?1):GraphStep@[a], Vertex(?4):VertexStep]
    } ,
    NeptuneTraverserConverterDFEStep
]
+ not converted into Neptune steps: HasStep([name.eq(josh)]),
Neptune steps:
[
    NeptuneInterleavingStep {
      StepInfo[joinVars=[?7, ?1], frontierElement=Vertex(?7):HasStep,
 pathElements={a=(last,Vertex(?1):GraphStep@[a])}, listPathElement={}, indexTime=0ms],
      DFEStep(Vertex) {
        DFENode {
          DFEJoinGroupNode[ children={
            DFEPatternNode[(?7, ?8, ?9, ?10) . project ALL[?7, ?9]
 graphFilters=(!= <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph> . ),
 {rangeCountEstimate=unknown}],
            DFEPatternNode[(?12, ?11, ?9, ?13) . project ALL[?9, ?12]
 graphFilters=(!= <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph> . ),
 {rangeCountEstimate=unknown}]
          }, {rangeCountEstimate=unknown}
          ]
        } [Vertex(?9):VertexStep, Vertex(?12):VertexStep]
      }
    }
]
+ not converted into Neptune steps: WherePredicateStep(eq(a)),
Neptune steps:
[
    DFECleanupStep
]


Optimized Traversal
===================
Neptune steps:
[
    DFEStep(Vertex) {
      DFENode {
        DFEJoinGroupNode[ children={
```

```
          DFEPatternNode[(?1, ?3, ?4, ?5) . project ALL[?1, ?4] graphFilters=(!=
 defaultGraph[526] . ), {rangeCountEstimate=9223372036854775807}]
        }, {rangeCountEstimate=unknown}
        ]
      } [Vertex(?1):GraphStep@[a], Vertex(?4):VertexStep]
    } ,
    NeptuneTraverserConverterDFEStep
]
+ not converted into Neptune steps: NeptuneHasStep([name.eq(josh)]),
Neptune steps:
[
    NeptuneMemoryTrackerStep,
    NeptuneInterleavingStep {
      StepInfo[joinVars=[?7, ?1], frontierElement=Vertex(?7):HasStep,
 pathElements={a=(last,Vertex(?1):GraphStep@[a])}, listPathElement={}, indexTime=0ms],
      DFEStep(Vertex) {
        DFENode {
          DFEJoinGroupNode[ children={
            DFEPatternNode[(?7, ?8, ?9, ?10) . project ALL[?7, ?9] graphFilters=(!=
 defaultGraph[526] . ), {rangeCountEstimate=9223372036854775807}],
            DFEPatternNode[(?12, ?11, ?9, ?13) . project ALL[?9, ?12] graphFilters=(!=
 defaultGraph[526] . ), {rangeCountEstimate=9223372036854775807}]
          }, {rangeCountEstimate=unknown}
          ]
        } [Vertex(?9):VertexStep, Vertex(?12):VertexStep]
      }
    }
]
+ not converted into Neptune steps: WherePredicateStep(eq(a)),
Neptune steps:
[
    DFECleanupStep
]


WARNING: >> [NeptuneHasStep([name.eq(josh)]), WherePredicateStep(eq(a))] << (or one of
 the children for each step) is not supported natively yet

Predicates
==========
# of predicates: 8
```

Consulte Informações no `explain` para obter uma descrição das seções específicas do DFE no relatório.

API **`profile`** do Gremlin no Neptune

A API `profile` do Gremlin do Neptune executa um percurso especificado do Gremlin, coleta várias métricas sobre a execução e produz um relatório de perfil como saída.

> **Note**
>
> Esse atributo está disponível a partir da Versão 1.0.1.0.200463.0 (15/10/2019).

Ela difere da etapa TinkerPop .profile () para poder relatar informações específicas do mecanismo Neptune.

O relatório de perfil inclui as seguintes informações sobre o plano de consulta:

- O pipeline do operador físico
- As operações de índice para execução e serialização de consultas
- O tamanho do resultado

A API `profile` usa uma versão estendida da sintaxe da API HTTP para consulta, com `/gremlin/profile` como o endpoint, em vez de `/gremlin`.

Parâmetros específicos **`profile`** do Gremlin no Neptune

- profile.results: `boolean`, valores permitidos: `TRUE` e `FALSE`, valor padrão: `TRUE`.

  Se verdadeiro, os resultados da consulta serão coletados e exibidos como parte do relatório de `profile`. Se falso, somente a contagem de resultados será exibida.
- profile.chop: `int`, valor padrão: 250.

  Se diferente de zero, fará com que a string de resultados seja truncada nesse número de caracteres. Isso não impede que todos os resultados sejam capturados. Ele simplesmente limita o tamanho da string no relatório de perfil. Se definido como zero, a string conterá todos os resultados.
- profile.serializer – `string`, valor padrão: `<null>`.

Se não forem nulos, os resultados coletados serão retornados em uma mensagem de resposta serializada no formato especificado por esse parâmetro. O número de operações de índice necessárias para produzir essa mensagem de resposta é relatado junto com o tamanho em bytes a serem enviados ao cliente.

Os valores permitidos são `<null>` ou qualquer um dos valores de enumeração “Serializadores” válidos do tipo MIME ou TinkerPop driver.

```
"application/json" or "GRAPHSON"
"application/vnd.gremlin-v1.0+json" or "GRAPHSON_V1"
"application/vnd.gremlin-v1.0+json;types=false" or "GRAPHSON_V1_UNTYPED"
"application/vnd.gremlin-v2.0+json" or "GRAPHSON_V2"
"application/vnd.gremlin-v2.0+json;types=false" or "GRAPHSON_V2_UNTYPED"
"application/vnd.gremlin-v3.0+json" or "GRAPHSON_V3"
"application/vnd.gremlin-v3.0+json;types=false" or "GRAPHSON_V3_UNTYPED"
"application/vnd.graphbinary-v1.0" or "GRAPHBINARY_V1"
```

- profile.indexOps: `boolean`, valores permitidos: `TRUE` e `FALSE`, valor padrão: `FALSE`.

  Se verdadeiro, mostra um relatório detalhado de todas as operações de índice que ocorreram durante a execução e a serialização da consulta. Aviso: esse relatório pode ser detalhado.

Exemplo de saída de **`profile`** do Gremlin no Neptune

Veja a seguir uma consulta de `profile` de exemplo.

```
curl -X POST https://your-neptune-endpoint:port/gremlin/profile \
     -d '{"gremlin":"g.V().hasLabel(\"airport\")
                         .has(\"code\", \"AUS\")
                         .emit()
                         .repeat(in().simplePath())
                         .times(2)
                         .limit(100)",
          "profile.serializer":"application/vnd.gremlin-v3.0+gryo"}'
```

Essa consulta gera o seguinte relatório de `profile` quando executada no gráfico de exemplo de rotas aéreas da postagem no blog, Let Me Graph That For You – Part 1 – Air Routes.

```
************************************************************
```

```
                Neptune Gremlin Profile
************************************************************

Query String
==================
g.V().hasLabel("airport").has("code",
 "AUS").emit().repeat(in().simplePath()).times(2).limit(100)

Original Traversal
==================
[GraphStep(vertex,[]), HasStep([~label.eq(airport), code.eq(AUS)]),
 RepeatStep(emit(true),[VertexStep(IN,vertex), PathFilterStep(simple),
 RepeatEndStep],until(loops(2))), RangeGlobalStep(0,100)]

Optimized Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <code>, "AUS", ?) . project ?1 .],
 {estimatedCardinality=1, indexTime=84, hashJoin=true, joinTime=3, actualTotalOutput=1}
            PatternNode[(?1, <~label>, ?2=<airport>, <~>) . project ask .],
 {estimatedCardinality=3374, indexTime=29, hashJoin=true, joinTime=0,
 actualTotalOutput=61}
            RepeatNode {
                Repeat {
                    PatternNode[(?3, ?5, ?1, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .
 SimplePathFilter(?1, ?3)) .], {hashJoin=true, estimatedCardinality=50148, indexTime=0,
 joinTime=3}
                }
                Emit {
                    Filter(true)
                }
                LoopsCondition {
                    LoopsFilter([?1, ?3],eq(2))
                }
            }, annotations={repeatMode=BFS, emitFirst=true, untilFirst=false, leftVar=?
1, rightVar=?3}
        }, finishers=[limit(100)], annotations={path=[Vertex(?1):GraphStep,
 Repeat[Vertex(?3):VertexStep]], joinStats=true, optimizationTime=495, maxVarId=7,
 executionTime=323}
    },
    NeptuneTraverserConverterStep
```

```
]

Physical Pipeline
=================
NeptuneGraphQueryStep
    |-- StartOp
    |-- JoinGroupOp
        |-- SpoolerOp(100)
        |-- DynamicJoinOp(PatternNode[(?1, <code>, "AUS", ?) . project ?1 .],
 {estimatedCardinality=1, indexTime=84, hashJoin=true})
        |-- SpoolerOp(100)
        |-- DynamicJoinOp(PatternNode[(?1, <~label>, ?2=<airport>, <~>) . project
 ask .], {estimatedCardinality=3374, indexTime=29, hashJoin=true})
        |-- RepeatOp
            |-- <upstream input> (Iteration 0) [visited=1, output=1 (until=0, emit=1),
 next=1]
            |-- BindingSetQueue (Iteration 1) [visited=61, output=61 (until=0,
 emit=61), next=61]
                |-- SpoolerOp(100)
                |-- DynamicJoinOp(PatternNode[(?3, ?5, ?1, ?6) . project ?
1,?3 . IsEdgeIdFilter(?6) . SimplePathFilter(?1, ?3)) .], {hashJoin=true,
 estimatedCardinality=50148, indexTime=0})
            |-- BindingSetQueue (Iteration 2) [visited=38, output=38 (until=38,
 emit=0), next=0]
                |-- SpoolerOp(100)
                |-- DynamicJoinOp(PatternNode[(?3, ?5, ?1, ?6) . project ?
1,?3 . IsEdgeIdFilter(?6) . SimplePathFilter(?1, ?3)) .], {hashJoin=true,
 estimatedCardinality=50148, indexTime=0})
        |-- LimitOp(100)

Runtime (ms)
============
Query Execution:  392.686
Serialization:   2636.380

Traversal Metrics
=================
Step                                                               Count  Traversers
    Time (ms)    % Dur
-------------------------------------------------------------------------------------
NeptuneGraphQueryStep(Vertex)                                        100         100
      314.162    82.78
NeptuneTraverserConverterStep                                        100         100
       65.333    17.22
```

```
                                            >TOTAL                     -           -
      379.495        -

Repeat Metrics
==============
Iteration  Visited   Output    Until     Emit     Next
----------------------------------------------------------
        0        1        1        0        1        1
        1       61       61        0       61       61
        2       38       38       38        0        0
----------------------------------------------------------
               100      100       38       62       62

Predicates
==========
# of predicates: 16

WARNING: reverse traversal with no edge label(s) - .in() / .both() may impact query
 performance

Results
=======
Count: 100
Output: [v[3], v[3600], v[3614], v[4], v[5], v[6], v[7], v[8], v[9], v[10], v[11],
 v[12], v[47], v[49], v[136], v[13], v[15], v[16], v[17], v[18], v[389], v[20], v[21],
 v[22], v[23], v[24], v[25], v[26], v[27], v[28], v[416], v[29], v[30], v[430], v[31],
 v[9...
Response serializer: GRYO_V3D0
Response size (bytes): 23566

Index Operations
================
Query execution:
    # of statement index ops: 3
    # of unique statement index ops: 3
    Duplication ratio: 1.0
    # of terms materialized: 0
Serialization:
    # of statement index ops: 200
    # of unique statement index ops: 140
    Duplication ratio: 1.43
    # of terms materialized: 393
```

Além dos planos de consulta gerados por uma chamada para `explain` do Neptune, os resultados de `profile` incluem as estatísticas de runtime sobre a execução da consulta. Cada operação Join é marcada com o tempo levado para realizar a junção, bem como com o número real de soluções que passaram por ela.

A saída de `profile` inclui o tempo levado durante a fase de execução da consulta principal, bem como a fase de serialização, se a opção `profile.serializer` tiver sido especificada.

O detalhamento das operações de índice realizadas durante cada fase também é incluído na parte inferior da saída de `profile`.

Observe que execuções consecutivas da mesma consulta podem mostrar resultados diferentes em termos de tempo de execução e operações de índice devido ao armazenamento em cache.

Para consultas que usam a etapa `repeat()`, um detalhamento da fronteira em cada iteração estará disponível, se a etapa `repeat()` tiver sido reduzida como parte de uma `NeptuneGraphQueryStep`.

Diferenças nos relatórios **`profile`** quando o DFE está habilitado

Quando o mecanismo de consulta alternativo do DFE do Neptune está habilitado, a saída de `profile` é um pouco diferente:

Percurso otimizado: essa seção é semelhante àquela da saída de `explain`, mas contém informações adicionais. Isso inclui o tipo de operadores do DFE que foram considerados no planejamento e as estimativas de custo associadas ao pior e ao melhor caso.

Pipeline físico: essa seção captura os operadores usados para executar a consulta. Elementos `DFESubQuery` abstraem o plano físico usado pelo DFE para executar a parte do plano pela qual ele é responsável. Os elementos `DFESubQuery` são desdobrados na seção a seguir, na qual as estatísticas do DFE são listadas.

QueryEngine Estatísticas do DFE: esta seção aparece somente quando pelo menos parte da consulta é executada pelo DFE. Ela descreve várias estatísticas de runtime específicas do DFE e contém uma análise detalhada do tempo gasto nas várias partes da execução da consulta, por `DFESubQuery`.

As subconsultas aninhadas em diferentes elementos `DFESubQuery` são niveladas nessa seção, e os identificadores exclusivos são marcados com um cabeçalho iniciado com `subQuery=`.

Métricas de percurso: essa seção mostra métricas de percurso em nível de etapa e, quando o mecanismo do DFE executa toda ou parte da consulta, exibe métricas para `DFEStep` e/ou `NeptuneInterleavingStep`. Consulte Ajustar consultas do Gremlin usando `explain` e `profile`.

> **Note**
>
> O DFE é um atributo experimental lançado no modo de laboratório, portanto, o formato exato da saída `profile` ainda está sujeito a alterações.

Exemplo de saída **`profile`** quando o mecanismo do Dataflow (DFE) do Neptune está habilitado.

Quando o mecanismo do DFE está sendo usado para executar consultas do Gremlin, a saída do API `profile` do Gremlin é formatada conforme mostrado no exemplo abaixo.

Consulta:

```
curl https://localhost:8182/gremlin/profile \
  -d "{\"gremlin\": \"g.withSideEffect('Neptune#useDFE', true).V().has('code',
 'ATL').out()\"}"
```

```
*******************************************************
                    Neptune Gremlin Profile
    *******************************************************

    Query String
    ==================
    g.withSideEffect('Neptune#useDFE', true).V().has('code', 'ATL').out()

    Original Traversal
    ==================
    [GraphStep(vertex,[]), HasStep([code.eq(ATL)]), VertexStep(OUT,vertex)]

    Optimized Traversal
    ===================
    Neptune steps:
    [
        DFEStep(Vertex) {
          DFENode {
            DFEJoinGroupNode[null](
              children=[
```

```
               DFEPatternNode((?1, vp://code[419430926], ?4, defaultGraph[526]) .
project DISTINCT[?1] objectFilters=(in(ATL[452987149]) . ), {rangeCountEstimate=1},
                 opInfo=(type=PipelineJoin,
cost=(exp=(in=1.00,out=1.00,io=0.00,comp=0.00,mem=0.00),wc=(in=1.00,out=1.00,io=0.00,comp=0.00
                   disc=(type=PipelineScan,
cost=(exp=(in=1.00,out=1.00,io=0.00,comp=0.00,mem=34.00),wc=(in=1.00,out=1.00,io=0.00,comp=0.0
               DFEPatternNode((?1, ?5, ?6, ?7) . project ALL[?1, ?6] graphFilters=(!=
defaultGraph[526] . ), {rangeCountEstimate=9223372036854775807})],
             opInfo=[
               OperatorInfoWithAlternative[
                 rec=(type=PipelineJoin,
cost=(exp=(in=1.00,out=27.76,io=0.00,comp=0.00,mem=0.00),wc=(in=1.00,out=27.76,io=0.00,comp=0.
                   disc=(type=PipelineScan,
cost=(exp=(in=1.00,out=27.76,io=Infinity,comp=0.00,mem=295147905179352830000.00),wc=(in=1.00,o
                 alt=(type=PipelineScan,
cost=(exp=(in=1.00,out=27.76,io=Infinity,comp=0.00,mem=295147905179352830000.00),wc=(in=1.00,o
         } [Vertex(?1):GraphStep, Vertex(?6):VertexStep]
       } ,
       NeptuneTraverserConverterDFEStep,
       DFECleanupStep
   ]


   Physical Pipeline
   =================
   DFEStep
       |-- DFESubQuery1

   DFEQueryEngine Statistics
   =================
   DFESubQuery1

#################################################################################################
   # ID # Out #1 # Out #2 # Name                  # Arguments
                                                                          # Mode #
Units In # Units Out # Ratio  # Time (ms) #

#################################################################################################
   # 0  # 1      # -      # DFESolutionInjection  # solutions=[]
                                                                        # -    # 0
     # 1         # 0.00   # 0.01      #
   #    #        #        #                       # outSchema=[]
                                                                        #      #
     #           #        #           #
```

```
 ################################################################################################
    # 1  # 2      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#089f43e3-4d71-4259-8d19-254ff63cee04/graph_1 # -    #
 1        # 1         # 1.00   # 0.02      #

 ################################################################################################
    # 2  # 3      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#089f43e3-4d71-4259-8d19-254ff63cee04/graph_2 # -    #
 1        # 242       # 242.00 # 0.02      #

 ################################################################################################
    # 3  # 4      # -      # DFEMergeChunks        # -
                                                                                  # -    # 242
      # 242       # 1.00   # 0.01      #

 ################################################################################################
    # 4  # -      # -      # DFEDrain              # -
                                                                                  # -    # 242
      # 0         # 0.00   # 0.01      #

 ################################################################################################


    subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/
graph#089f43e3-4d71-4259-8d19-254ff63cee04/graph_1

 ################################################################################################
    # ID # Out #1 # Out #2 # Name                 # Arguments
                          # Mode # Units In # Units Out # Ratio # Time (ms) #

 ################################################################################################
    # 0  # 1      # -      # DFEPipelineScan      # pattern=Node(?1) with property
 'code' as ?4 and label 'ALL' # -    # 0        # 1         # 0.00  # 0.22      #
    #    #        #        #                     # inlineFilters=[(?4 IN ["ATL"])]
                          #      #          #           #       #           #
    #    #        #        #                     # patternEstimate=1
                          #      #          #           #       #           #

 ################################################################################################
    # 1  # 2      # -      # DFEMergeChunks       # -
                          # -    # 1        # 1         # 1.00  # 0.02      #

 ################################################################################################
```

```
    # 2  # 4      # -      # DFERelationalJoin    # joinVars=[]
                          # -    # 2        # 1         # 0.50  # 0.09      #

 ##########################################################################################
    # 3  # 2      # -      # DFESolutionInjection # solutions=[]
                          # -    # 0        # 1         # 0.00  # 0.01      #
    #    #        #        #                      # outSchema=[]
                          #      #          #           #       #           #

 ##########################################################################################
    # 4  # -      # -      # DFEDrain             # -
                          # -    # 1        # 0         # 0.00  # 0.01      #

 ##########################################################################################


    subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/
graph#089f43e3-4d71-4259-8d19-254ff63cee04/graph_2

 ##########################################################################################
    # ID # Out #1 # Out #2 # Name                # Arguments
 # Mode # Units In # Units Out # Ratio  # Time (ms) #

 ##########################################################################################
    # 0  # 1      # -      # DFESolutionInjection # solutions=[]
 # -    # 0        # 1         # 0.00   # 0.01      #
    #    #        #        #                      # outSchema=[?1]
 #      #          #           #        #           #

 ##########################################################################################
    # 1  # 2      # 3      # DFETee               # -
 # -    # 1        # 2         # 2.00   # 0.01      #

 ##########################################################################################
    # 2  # 4      # -      # DFEDistinctColumn    # column=?1
 # -    # 1        # 1         # 1.00   # 0.21      #
    #    #        #        #                      # ordered=false
 #      #          #           #        #           #

 ##########################################################################################
    # 3  # 5      # -      # DFEHashIndexBuild    # vars=[?1]
 # -    # 1        # 1         # 1.00   # 0.03      #

 ##########################################################################################
```

```
   # 4  # 5      # -      # DFEPipelineJoin      # pattern=Edge((?1)-[?7:?5]->(?6))
# -    # 1        # 242       # 242.00 # 0.51      #
   #    #        #        #                      # constraints=[]
#      #          #          #        #          #
   #    #        #        #                      # patternEstimate=9223372036854775807
#      #          #          #        #          #

#################################################################################################
   # 5  # 6      # 7      # DFESync              # -
# -    # 243      # 243       # 1.00   # 0.02      #

#################################################################################################
   # 6  # 8      # -      # DFEForwardValue      # -
# -    # 1        # 1         # 1.00   # 0.01      #

#################################################################################################
   # 7  # 8      # -      # DFEForwardValue      # -
# -    # 242      # 242       # 1.00   # 0.02      #

#################################################################################################
   # 8  # 9      # -      # DFEHashIndexJoin     # -
# -    # 243      # 242       # 1.00   # 0.31      #

#################################################################################################
   # 9  # -      # -      # DFEDrain             # -
# -    # 242      # 0         # 0.00   # 0.01      #

#################################################################################################


   Runtime (ms)
   ============
   Query Execution: 11.744

   Traversal Metrics
   =================
   Step                                                               Count
Traversers       Time (ms)    % Dur

-------------------------------------------------------------------------------------------------
   DFEStep(Vertex)                                                    242
242          10.849    95.48
   NeptuneTraverserConverterDFEStep                                   242
242           0.514     4.52
```

```
                                                >TOTAL                         -
-          11.363        -

   Predicates
   ==========
   # of predicates: 18

   Results
   =======
   Count: 242


   Index Operations
   ================
   Query execution:
       # of statement index ops: 0
       # of terms materialized: 0
```

> **Note**
>
> O DFE é um atributo experimental lançado no modo de laboratório, portanto, o formato exato da saída `profile` ainda está sujeito a alterações.

Ajustar consultas do Gremlin usando **`explain`** e **`profile`**

Muitas vezes, é possível ajustar as consultas do Gremlin no Amazon Neptune para obter um melhor desempenho, usando as informações disponíveis nos relatórios obtidos das APIs explain e profile do Neptune. Para isso, é útil entender como o Neptune processa os percursos do Gremlin.

> **Important**
>
> Foi feita uma alteração na TinkerPop versão 3.4.11 que melhora a exatidão de como as consultas são processadas, mas, no momento, às vezes pode afetar seriamente o desempenho das consultas.
> Por exemplo, uma consulta desse tipo pode apresentar uma lentidão significativa:
>
> ```
> g.V().hasLabel('airport').
>   order().
>     by(out().count(),desc).
>   limit(10).
> ```

```
  out()
```

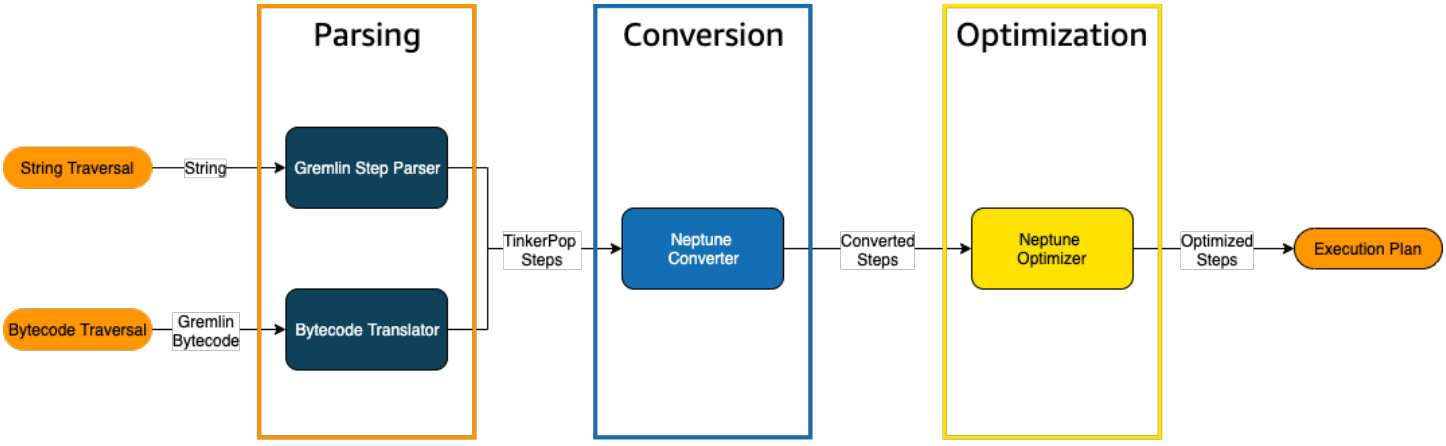
Os vértices após a etapa limite agora são buscados de uma forma não ideal por causa da alteração 3.4.11. TinkerPop Para evitar isso, é possível modificar a consulta adicionando a etapa barrier() a qualquer momento após `order().by()`. Por exemplo: .

```
g.V().hasLabel('airport').
  order().
    by(out().count(),desc).
  limit(10).
  barrier().
  out()
```

TinkerPop 3.4.11 foi habilitado na versão 1.0.5.0 do motor Neptune.

Noções básicas sobre o processamento de percursos do Gremlin no Neptune

Quando um percurso do Gremlin é enviado ao Neptune, há três processos principais que transformam o percurso em um plano de execução subjacente a ser executado pelo mecanismo. São eles: análise, conversão e otimização:



O processo de análise de percurso

A primeira etapa do processamento de um percurso é analisá-lo em uma linguagem comum. Em Neptune, essa linguagem comum é o conjunto TinkerPop de etapas que fazem parte da API. TinkerPop Cada uma dessas etapas representa uma unidade de cálculo dentro do percurso.

É possível enviar um percurso do Gremlin ao Neptune como uma string ou um bytecode. O endpoint REST e o método `submit()` do driver do cliente Java enviam percursos como strings, como neste exemplo:

```
client.submit("g.V()")
```

Aplicações e drivers de linguagem que usam variantes de linguagem Gremlin (GLV) enviam percursos em bytecode.

O processo de conversão de percurso

A segunda etapa no processamento de uma travessia é converter suas TinkerPop etapas em um conjunto de etapas de Netuno convertidas e não convertidas. A maioria das etapas na linguagem de consulta Apache TinkerPop Gremlin é convertida em etapas específicas do Neptune que são otimizadas para serem executadas no mecanismo Neptune subjacente. Quando uma TinkerPop etapa sem um equivalente de Netuno é encontrada em uma travessia, essa etapa e todas as etapas subsequentes na travessia são processadas pelo mecanismo de consulta. TinkerPop

Para obter mais informações sobre quais etapas podem ser convertidas em quais circunstâncias, consulte Suporte a etapas do Gremlin.

O processo de otimização de percurso

A etapa final do processamento de percurso é executar a série de etapas convertidas e não convertidas por meio do otimizador, para tentar determinar o melhor plano de execução. O resultado dessa otimização é o plano de execução que o mecanismo do Neptune processa.

Usar a API **`explain`** do Gremlin no Neptune para ajustar consultas

A API de explicação do Neptune não é a mesma que a etapa `explain()` do Gremlin. Ela retorna o plano de execução final que o mecanismo do Neptune processaria ao executar a consulta. Como não executa nenhum processamento, ela retorna o mesmo plano, independentemente dos parâmetros usados, e sua saída não contém estatísticas sobre a execução real.

Examine o seguinte percurso simples que encontra todos os vértices do aeroporto para Anchorage:

```
g.V().has('code','ANC')
```

É possível executar esse percurso por meio da API `explain` do Neptune de duas maneiras. A primeira é fazer uma chamada REST para o endpoint de explicação, desta forma:

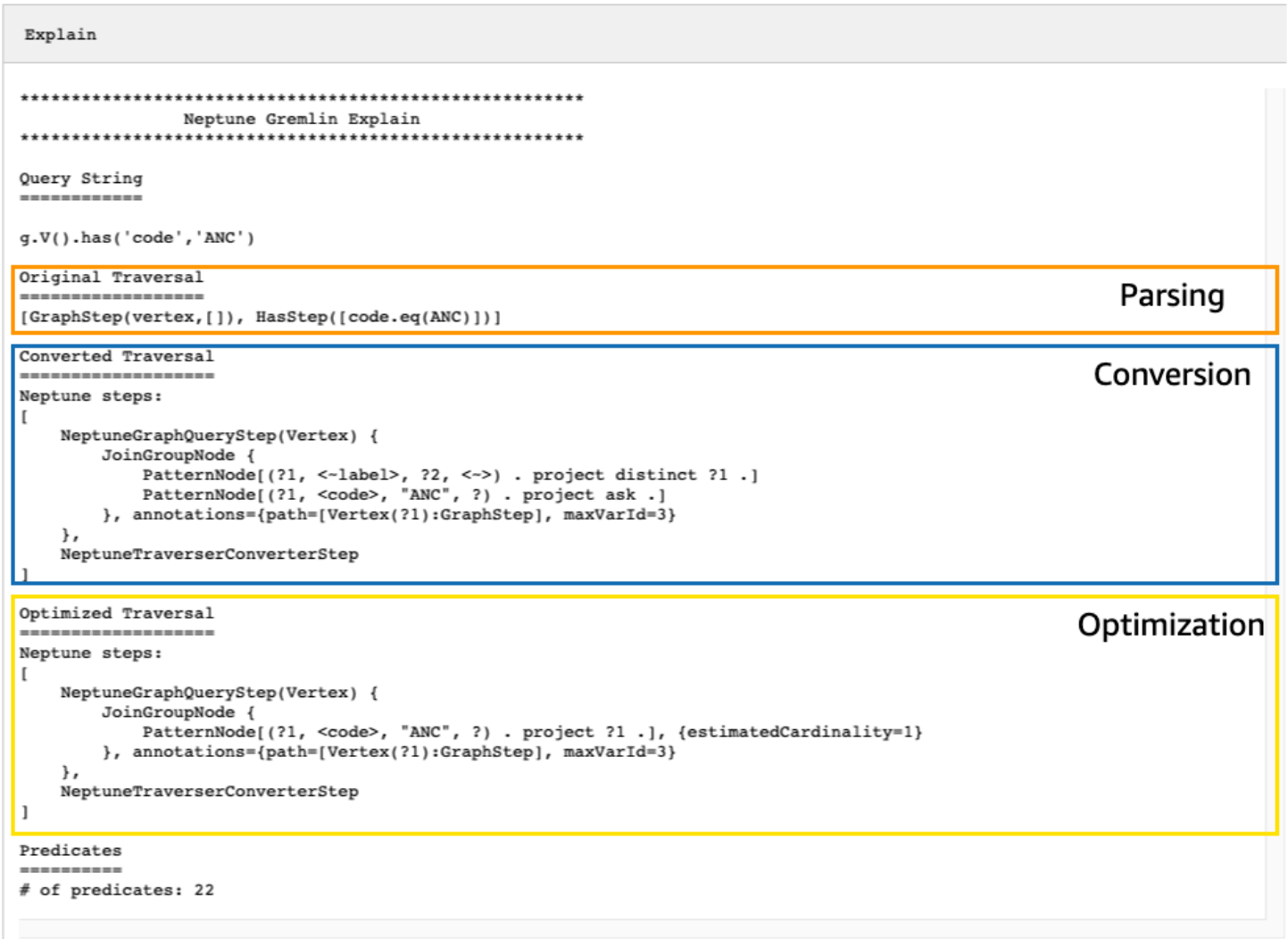
```
curl -X POST https://your-neptune-endpoint:port/gremlin/explain -d
 '{"gremlin":"g.V().has('code','ANC')"}'
```

A segunda é usar a magia de célula %%gremlin da bancada de trabalho do Neptune com o parâmetro `explain`. Isso transmite o percurso contido no corpo da célula para a API `explain` do Neptune e, depois, exibe a saída resultante quando você executa a célula:

```
%%gremlin explain

g.V().has('code','ANC')
```

A saída da API `explain` resultante descreve o plano de execução do Neptune para o percurso. Como você pode ver na imagem abaixo, o plano inclui cada uma das três etapas no pipeline de processamento:



Ajustar um percurso observando as etapas que não são convertidas

Um dos primeiros itens a procurar na saída da API `explain` do Neptune são as etapas do Gremlin que não são convertidas em etapas nativas do Neptune. Em um plano de consulta, quando é encontrada uma etapa que não pode ser convertida em uma etapa nativa do Neptune, ela e todas as etapas subsequentes do plano são processadas pelo servidor do Gremlin.

No exemplo acima, todas as etapas no percurso foram convertidas. Vamos examinar a saída da API `explain` para este percurso:

```
g.V().has('code','ANC').out().choose(hasLabel('airport'), values('code'), constant('Not
 an airport'))
```

Como você pode ver na imagem abaixo, o Neptune não conseguiu converter a etapa `choose()`:

Explain

```
*****************************************************
               Neptune Gremlin Explain
*****************************************************

Query String
============

g.V().has('code','ANC').out().choose(hasLabel('airport'), values('code'), constant('Not an airport'))

Original Traversal
==================
[GraphStep(vertex,[]), HasStep([code.eq(ANC)]), VertexStep(OUT,vertex), ChooseStep([HasStep([~label.eq(airport)]), HasNex

Converted Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .]
            PatternNode[(?1, <code>, "ANC", ?) . project ask .]
            PatternNode[(?1, ?5, ?3, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .]
            PatternNode[(?3, <~label>, ?4, <~>) . project ask .]
        }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep], maxVarId=7}
    },
    NeptuneTraverserConverterStep
]
+ not converted into Neptune steps: [ChooseStep([HasStep([~label.eq(airport)]), HasNextStep],[[(eq(true)), [PropertiesSte

Optimized Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <code>, "ANC", ?) . project ?1 .], {estimatedCardinality=1}
            PatternNode[(?1, ?5, ?3, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .], {estimatedCardinality=INFINITY}
        }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep], maxVarId=7}
    },
    NeptuneTraverserConverterStep
]
+ not converted into Neptune steps: [ChooseStep([NeptuneHasStep([~label.eq(airport)]), HasNextStep],[[(eq(true)), [Proper

WARNING: >> ChooseStep([NeptuneHasStep([~label.eq(airport)]), HasNextStep],[[(eq(true)), [PropertiesStep([code],value), E

Predicates
==========
# of predicates: 26
```

É possível executar algumas etapas para ajustar o desempenho do percurso. A primeira seria reescrevê-lo de forma a eliminar a etapa que não pôde ser convertida. Outra seria mover a etapa para o final do percurso para que todas as outras etapas possam ser convertidas em etapas nativas.

Um plano de consulta com etapas que não são convertidas nem sempre precisa ser ajustado. Se as etapas que não podem ser convertidas estiverem no final do percurso e estiverem relacionadas à forma como a saída é formatada e não à forma como o grafo é percorrido, elas poderão ter pouco efeito no desempenho.

Outro item a ser observado ao examinar a saída da API `explain` do Neptune são as etapas que não usam índices. O seguinte percurso encontra todos os aeroportos com voos que aterrissam em Anchorage:

```
g.V().has('code','ANC').in().values('code')
```

A saída da API de explicação para esse percurso é:

```
*******************************************************
                Neptune Gremlin Explain
*******************************************************

Query String
============

g.V().has('code','ANC').in().values('code')

Original Traversal
==================
[GraphStep(vertex,[]), HasStep([code.eq(ANC)]), VertexStep(IN,vertex),
 PropertiesStep([code],value)]

Converted Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(PropertyValue) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .]
            PatternNode[(?1, <code>, "ANC", ?) . project ask .]
            PatternNode[(?3, ?5, ?1, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .]
            PatternNode[(?3, <~label>, ?4, <~>) . project ask .]
            PatternNode[(?3, ?7, ?8, <~>) . project ?3,?8 . ContainsFilter(?7 in
 (<code>)) .]
       }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep,
 PropertyValue(?8):PropertiesStep], maxVarId=9}
    },
    NeptuneTraverserConverterStep
]

Optimized Traversal
===================
```

```
Neptune steps:
[
    NeptuneGraphQueryStep(PropertyValue) {
        JoinGroupNode {
            PatternNode[(?1, <code>, "ANC", ?) . project ?1 .],
 {estimatedCardinality=1}
            PatternNode[(?3, ?5, ?1, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .],
 {estimatedCardinality=INFINITY}
            PatternNode[(?3, ?7=<code>, ?8, <~>) . project ?3,?8 .],
 {estimatedCardinality=7564}
       }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep,
 PropertyValue(?8):PropertiesStep], maxVarId=9}
    },
    NeptuneTraverserConverterStep
]

Predicates
==========
# of predicates: 26

WARNING: reverse traversal with no edge label(s) - .in() / .both() may impact query
 performance
```

A mensagem `WARNING` na parte inferior da saída ocorre porque a etapa `in()` no percurso não pode ser processada usando um dos três índices mantidos pelo Neptune (consulte Como as declarações são indexadas no Neptune e Declarações do Gremlin no Neptune). Como a etapa `in()` não contém filtro de borda, ela não pode ser resolvida usando o índice `SPOG`, `POGS` ou `GPSO`. Em vez disso, o Neptune deve realizar uma verificação de união para encontrar os vértices solicitados, o que é muito menos eficiente.

Há duas maneiras de ajustar o percurso nessa situação. A primeira é adicionar um ou mais critérios de filtragem à etapa `in()` para que uma pesquisa indexada possa ser usada para resolver a consulta. Para o exemplo acima, pode ser:

```
g.V().has('code','ANC').in('route').values('code')
```

A saída da API `explain` do Neptune para o percurso revisado não contém mais a mensagem `WARNING`:

```
*******************************************************
                Neptune Gremlin Explain
```

```
*******************************************************

Query String
============

g.V().has('code','ANC').in('route').values('code')

Original Traversal
==================
[GraphStep(vertex,[]), HasStep([code.eq(ANC)]), VertexStep(IN,[route],vertex),
 PropertiesStep([code],value)]

Converted Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(PropertyValue) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .]
            PatternNode[(?1, <code>, "ANC", ?) . project ask .]
            PatternNode[(?3, ?5, ?1, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .
 ContainsFilter(?5 in (<route>)) .]
            PatternNode[(?3, <~label>, ?4, <~>) . project ask .]
            PatternNode[(?3, ?7, ?8, <~>) . project ?3,?8 . ContainsFilter(?7 in
 (<code>)) .]
        }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep,
 PropertyValue(?8):PropertiesStep], maxVarId=9}
    },
    NeptuneTraverserConverterStep
]

Optimized Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(PropertyValue) {
        JoinGroupNode {
            PatternNode[(?1, <code>, "ANC", ?) . project ?1 .],
 {estimatedCardinality=1}
            PatternNode[(?3, ?5=<route>, ?1, ?6) . project ?1,?3 . IsEdgeIdFilter(?
6) .], {estimatedCardinality=32042}
            PatternNode[(?3, ?7=<code>, ?8, <~>) . project ?3,?8 .],
 {estimatedCardinality=7564}
```

```
        }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep,
 PropertyValue(?8):PropertiesStep], maxVarId=9}
    },
    NeptuneTraverserConverterStep
]

Predicates
==========
# of predicates: 26
```

Outra opção, se você estiver executando muitos percursos desse tipo, é executá-los em um cluster de banco de dados do Neptune que tenha o índice opcional OSGP habilitado (consulte Habilitar um índice OSGP). Habilitar um índice OSGP tem desvantagens:

- Ele deve ser habilitado em um cluster de banco de dados antes do carregamento de dados.
- As taxas de inserção para vértices e bordas podem diminuir em até 23%.
- O uso do armazenamento aumentará em cerca de 20%.
- Consultas de leitura que dispersam solicitações em todos os índices podem ter latências maiores.

Ter um índice OSGP faz muito sentido para um conjunto restrito de padrões de consulta, mas, a menos que você os execute com frequência, geralmente é preferível tentar garantir que os percursos escritos possam ser resolvidos usando os três índices principais.

Usar um grande número de predicados

O Neptune trata cada rótulo de borda e cada nome distinto de propriedade de vértice ou borda no grafo como um predicado e foi projetado por padrão para funcionar com um número relativamente baixo de predicados distintos. Quando você tem mais do que alguns milhares de predicados em seus dados de grafo, o desempenho pode diminuir.

A saída `explain` do Neptune avisará se for esse o caso:

```
Predicates
==========
# of predicates: 9549
WARNING: high predicate count (# of distinct property names and edge labels)
```

Se não for conveniente revisar o modelo de dados para reduzir o número de rótulos e propriedades e, portanto, o número de predicados, a melhor maneira de ajustar os percursos é executá-los em um cluster de banco de dados com o índice OSGP habilitado, conforme abordado acima.

Usar a API **`profile`** do Gremlin no Neptune para ajustar percursos

A API `profile` do Neptune é bem diferente da etapa `profile()` do Gremlin. Assim como a API `explain`, sua saída inclui o plano de consulta que o mecanismo do Neptune usa ao executar o percurso. Além disso, a saída `profile` inclui estatísticas reais de execução do percurso, considerando como seus parâmetros são definidos.

Novamente, considere o percurso simples que encontra todos os vértices do aeroporto para Anchorage:

```
g.V().has('code','ANC')
```

Assim como na API `explain`, é possível invocar a API `profile` usando uma chamada REST:

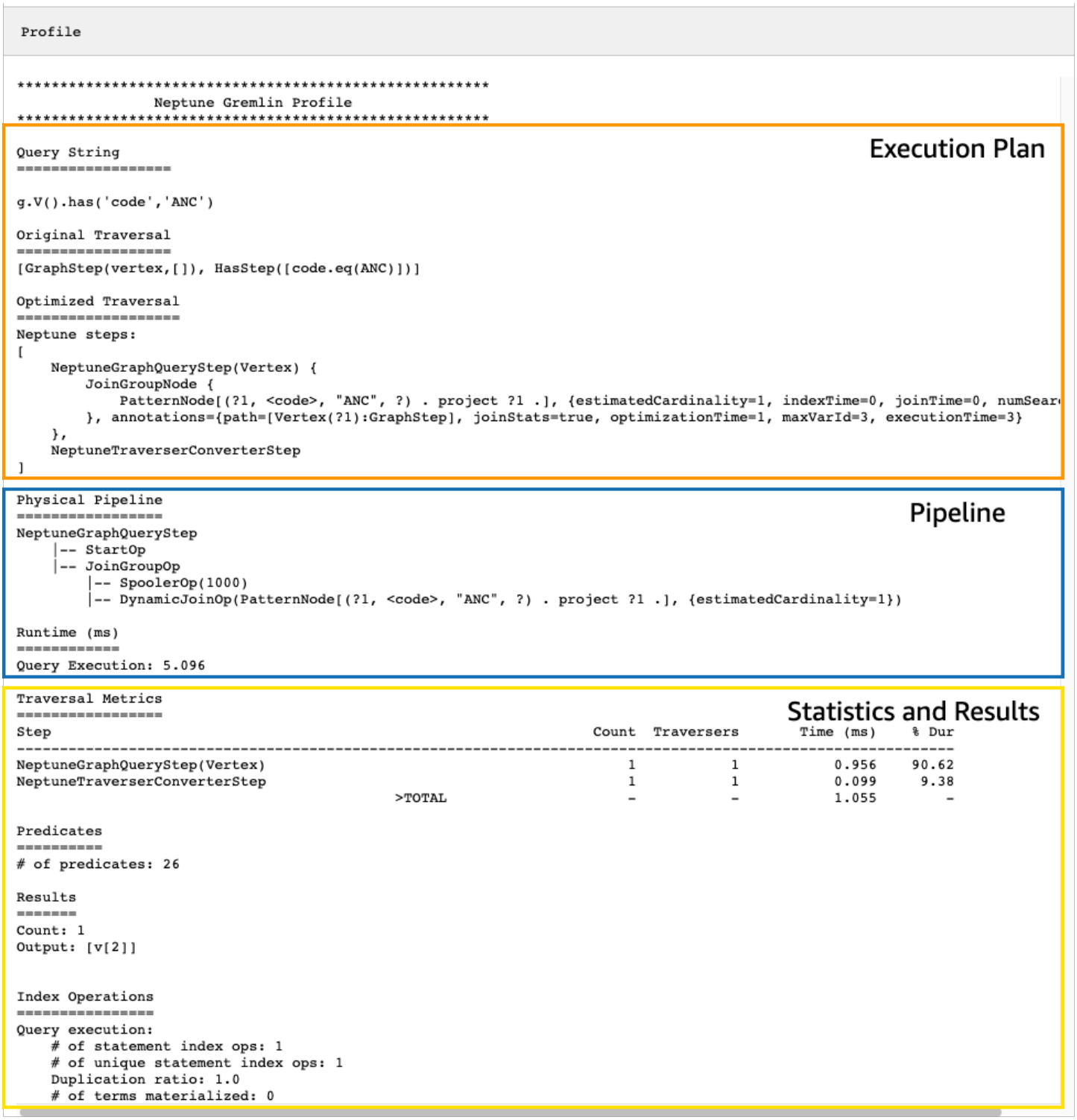
```
curl -X POST https://your-neptune-endpoint:port/gremlin/profile -d
 '{"gremlin":"g.V().has('code','ANC')"}'
```

Também é possível usar a magia de célula %%gremlin da bancada de trabalho do Neptune com o parâmetro `profile`. Isso transmite o percurso contido no corpo da célula para a API `profile` do Neptune e, depois, exibe a saída resultante quando você executa a célula:

```
%%gremlin profile

g.V().has('code','ANC')
```

A saída da API `profile` resultante contém o plano de execução do Neptune para o percurso e estatísticas sobre a execução do plano, como você pode ver nesta imagem:



Na saída `profile`, a seção do plano de execução contém somente o plano de execução final para o percurso, não as etapas intermediárias. A seção do pipeline contém as operações físicas executadas, bem como o tempo real (em milissegundos) que a execução do percurso levou. A

métrica de runtime é extremamente útil para comparar os tempos que duas versões diferentes de um percurso levam à medida que você as otimiza.

> **Note**
>
> O runtime inicial de um percurso geralmente é maior do que os runtimes subsequentes, porque o primeiro faz com que os dados relevantes sejam armazenados em cache.

A terceira seção da saída `profile` contém estatísticas de execução e os resultados do percurso. Para ver como essas informações podem ser úteis para ajustar um percurso, considere o seguinte percurso, que encontra todos os aeroportos cujo nome começa com “Anchora” e todos os aeroportos acessíveis em dois saltos a partir desses aeroportos, retornando códigos de aeroporto, rotas de voo e distâncias:

```
%%gremlin profile

g.withSideEffect("Neptune#fts.endpoint", "{your-OpenSearch-endpoint-URL").
    V().has("city", "Neptune#fts Anchora~").
    repeat(outE('route').inV().simplePath()).times(2).
    project('Destination', 'Route').
        by('code').
        by(path().by('code').by('dist'))
```

Métricas de percurso na saída da API **`profile`** do Neptune

O primeiro conjunto de métricas que está disponível em todas as saídas `profile` são as métricas de percurso. Elas são semelhantes às métricas de etapa `profile()` do Gremlin, com algumas diferenças:

```
Traversal Metrics
=================
Step                                                               Count  Traversers
    Time (ms)    % Dur
-------------------------------------------------------------------------------------------------
NeptuneGraphQueryStep(Vertex)                                       3856        3856
       91.701     9.09
NeptuneTraverserConverterStep                                       3856        3856
       38.787     3.84
ProjectStep([Destination, Route],[value(code), ...                  3856        3856
      878.786    87.07
```

```
  PathStep([value(code), value(dist)])                                3856        3856
     601.359
                                                >TOTAL                    -           -
    1009.274        -
```

A primeira coluna da tabela de métricas de percurso lista as etapas executadas pelo percurso. As duas primeiras etapas são geralmente as etapas específicas do Neptune, `NeptuneGraphQueryStep` e `NeptuneTraverserConverterStep`.

`NeptuneGraphQueryStep` representa o tempo de execução de toda a parte do percurso que poderia ser convertida e executada nativamente pelo mecanismo do Neptune.

`NeptuneTraverserConverterStep`representa o processo de conversão da saída dessas etapas convertidas em TinkerPop percursos que permitem que etapas que não puderam ser convertidas, se houver, sejam processadas ou retornem os resultados em um TinkerPop formato compatível.

No exemplo acima, temos várias etapas não convertidas, então vemos que cada uma dessas TinkerPop etapas (`ProjectStep`,`PathStep`) aparece como uma linha na tabela.

A segunda coluna na tabela,`Count`, relata o número de atravessadores representados que passaram pela etapa, enquanto a terceira coluna,`Traversers`, relata o número de atravessadores que passaram por essa etapa, conforme explicado na documentação da etapa do TinkerPop perfil.

Em nosso exemplo, há 3.856 vértices e 3.856 percursos gerados pela `NeptuneGraphQueryStep`, e esses números permanecem os mesmos durante todo o processamento restante porque `ProjectStep` e `PathStep` estão formatando os resultados, não os filtrando.

> **Note**
>
> Ao contrário TinkerPop disso, o motor Neptune não otimiza o desempenho aumentando suas etapas. `NeptuneGraphQueryStep` `NeptuneTraverserConverterStep` O aumento de volume é a TinkerPop operação que combina travessas no mesmo vértice para reduzir a sobrecarga operacional, e é isso que faz com que os `Count` números e sejam diferentes. `Traversers` Como o aumento de volume ocorre apenas nas etapas às quais Netuno delega, e não nas etapas que TinkerPop o Netuno manipula nativamente, as colunas e raramente diferem. `Count` `Traverser`

A coluna Tempo informa o número de milissegundos que a etapa levou, e a coluna `% Dur` informa qual a porcentagem do tempo total de processamento da etapa. Essas são as métricas que indicam onde concentrar seus esforços de ajuste, mostrando as etapas que levaram mais tempo.

Métricas de operação de índice na saída da API **`profile`** do Neptune

Outro conjunto de métricas na saída da API do perfil do Neptune são as operações de indexação:

```
Index Operations
================
Query execution:
    # of statement index ops: 23191
    # of unique statement index ops: 5960
    Duplication ratio: 3.89
    # of terms materialized: 0
```

Elas relatam:

- O número total de pesquisas de índice.
- O número de pesquisas de índice exclusivas realizadas.
- A proporção entre o total de pesquisas de índices e as exclusivas. Uma proporção menor indica menor redundância.
- O número de termos materializados do dicionário de termos.

Métricas repetidas na saída da API **`profile`** do Neptune

Se o percurso usar uma etapa `repeat()` como no exemplo acima, uma seção que contém métricas repetidas será exibida na saída `profile`:

```
Repeat Metrics
==============
Iteration  Visited   Output    Until     Emit     Next
-------------------------------------------------------
        0        2        0        0        0        2
        1       53        0        0        0       53
        2     3856     3856     3856        0        0
-------------------------------------------------------
              3911     3856     3856        0       55
```

Elas relatam:

- A contagem de loops de uma linha (a coluna `Iteration`).
- O número de elementos visitados pelo loop (a coluna `Visited`).
- O número de elementos gerados pelo loop (a coluna `Output`).
- O último elemento gerado pelo loop (a coluna `Until`).
- O número de elementos emitidos pelo loop (a coluna `Emit`).
- O número de elementos transmitidos do loop para o loop subsequente (a `Next` coluna).

Essas métricas repetidas são muito úteis para entender o fator de ramificação do percurso, para ter uma ideia de quanto trabalho está sendo feito pelo banco de dados. Você pode usar esses números para diagnosticar problemas de desempenho, especialmente quando o mesmo percurso tem um desempenho drasticamente diferente com parâmetros distintos.

Métricas de pesquisa de texto completo na saída da API **`profile`** do Neptune

Quando um percurso usa uma pesquisa de texto completo, como no exemplo acima, uma seção que contém as métricas de pesquisa de texto completo (FTS) aparece na saída `profile`:

```
FTS Metrics
==============
SearchNode[(idVar=?1, query=Anchora~, field=city) . project ?1 .],
    {endpoint=your-OpenSearch-endpoint-URL, incomingSolutionsThreshold=1000,
 estimatedCardinality=INFINITY,
    remoteCallTimeSummary=[total=65, avg=32.500000, max=37, min=28],
    remoteCallTime=65, remoteCalls=2, joinTime=0, indexTime=0, remoteResults=2}

    2 result(s) produced from SearchNode above
```

Isso mostra a consulta enviada ao cluster ElasticSearch (ES) e relata várias métricas sobre a interação com ElasticSearch que podem ajudá-lo a identificar problemas de desempenho relacionados à pesquisa de texto completo:

- Informações resumidas sobre as chamadas para o ElasticSearch índice:
  - O número total de milissegundos exigido por todas as chamadas remotas para atender à consulta (`total`).
  - O número médio de milissegundos gastos em uma remoteCall (`avg`).
  - O número mínimo de milissegundos gastos em uma remoteCall (`min`).
  - O número máximo de milissegundos gastos em uma remoteCall (`max`).

- Tempo total consumido por chamadas remotas para ElasticSearch ()`remoteCallTime`.
- O número de chamadas remotas feitas para ElasticSearch ()`remoteCalls`.
- O número de milissegundos gastos em junções de ElasticSearch resultados ()`joinTime`.
- O número de milissegundos gastos em pesquisas de índice (`indexTime`).
- O número total de resultados retornados por ElasticSearch (`remoteResults`).

Suporte nativo para etapas do Gremlin no Amazon Neptune

No momento, o mecanismo do Amazon Neptune não tem suporte nativo completo para todas as etapas do Gremlin, conforme explicado em Ajustar consultas do Gremlin. O suporte atual se divide em quatro categorias:

- Etapas do Gremlin que sempre podem ser convertidas em operações nativas do mecanismo do Neptune
- Etapas do Gremlin que sempre podem ser convertidas em operações nativas do mecanismo do Neptune em alguns casos
- Etapas do Gremlin que nunca são convertidas em operações nativas do mecanismo do Neptune
- Etapas do Gremlin não compatíveis com o Neptune

Etapas do Gremlin que sempre podem ser convertidas em operações nativas do mecanismo do Neptune

Muitas etapas do Gremlin podem ser convertidas em operações nativas do mecanismo do Neptune, desde que atendam às seguintes condições:

- Elas não são precedidas na consulta por uma etapa que não pode ser convertida.
- A etapa principal, se houver, pode ser convertida.
- Todos os percursos secundários, se houver, podem ser convertidos.

As seguintes etapas do Gremlin podem ser convertidas em operações nativas do mecanismo do Neptune se elas atenderem a essas condições:

- and( )
- as( )
- count( )

- E( )
- emit( )
- explain( )
- group( )
- groupCount( )
- has( )
- identity( )
- is( )
- key( )
- label( )
- limit( )
- local( )
- loops( )
- not( )
- or( )
- profile( )
- properties( )
- subgraph( )
- until( )
- V( )
- value( )
- valueMap( )
- values( )

Etapas do Gremlin que sempre podem ser convertidas em operações nativas do mecanismo do Neptune em alguns casos

Algumas etapas do Gremlin podem ser convertidas em operações nativas do mecanismo do Neptune em algumas situações, mas não em outras:

- addE (): a etapa `addE()` geralmente pode ser convertida em uma operação nativa do mecanismo do Neptune, a menos que seja imediatamente seguida por uma etapa `property()` que contenha um percurso como chave.

- addV (): a etapa `addV()` geralmente pode ser convertida em uma operação nativa do mecanismo do Neptune, a menos que seja imediatamente seguida por uma etapa `property()` que contenham um percurso como chave ou vários rótulos sejam atribuídos.
- aggregate (): a etapa `aggregate()` geralmente pode ser convertida em uma operação nativa do mecanismo do Neptune, a menos que a etapa seja usada em um percurso secundário ou subpercurso, ou a menos que o valor armazenado seja algo diferente de um vértice, uma borda, um ID, um rótulo ou um valor de propriedade.

  No exemplo abaixo, `aggregate()` não é convertido porque está sendo usado em um percurso secundário:

  ```
  g.V().has('code','ANC').as('a')
       .project('flights').by(select('a')
       .outE().aggregate('x'))
  ```

  Nesse exemplo, aggregate() não é convertido porque o item armazenado é o `min()` de um valor:

  ```
  g.V().has('code','ANC').outE().aggregate('x').by(values('dist').min())
  ```

- barrier( ): a etapa `barrier()` geralmente pode ser convertida em uma operação nativa do mecanismo do Neptune, a menos que a etapa seguinte não seja convertida.
- cap( ): o único caso em que a etapa `cap()` é convertida é quando ela é combinada com a etapa `unfold()` para gerar uma versão desdobrada de um agregado de valores de vértice, borda, ID ou propriedade. Neste exemplo, `cap()` será convertido porque é seguido por `.unfold()`:

  ```
  g.V().has('airport','country','IE').aggregate('airport').limit(2)
       .cap('airport').unfold()
  ```

  No entanto, se você remover o `.unfold()`, `cap()` não será convertido:

  ```
  g.V().has('airport','country','IE').aggregate('airport').limit(2)
       .cap('airport')
  ```

- coalesce () — O único caso em que a `coalesce()` etapa é convertida é quando ela segue o padrão Upsert recomendado na página de receitas. TinkerPop Outros padrões coalesce() não são permitidos. A conversão é limitada ao caso em que todos os percursos secundários podem ser convertidos, todos eles produzem o mesmo tipo de saída (vértice, borda, ID, valor, chave ou rótulo), percorrem um novo elemento e não contêm a etapa `repeat()`.

- constant( ): a etapa constant() no momento só será convertida se for usada em uma parte `sack().by()` de um percurso para atribuir um valor constante, como este:

```
g.V().has('code','ANC').sack(assign).by(constant(10)).out().limit(2)
```

- cyclicPath( ): em geral, a etapa `cyclicPath()` pode ser convertida em uma operação nativa do mecanismo do Neptune, a menos que ela seja usada com moduladores `by()`, `from()` ou `to()`. Nas seguintes consultas, por exemplo, `cyclicPath()` não é convertida:

```
g.V().has('code','ANC').as('a').out().out().cyclicPath().by('code')
g.V().has('code','ANC').as('a').out().out().cyclicPath().from('a')
g.V().has('code','ANC').as('a').out().out().cyclicPath().to('a')
```

- drop( ): em geral, a etapa `drop()` pode ser convertida em uma operação nativa do mecanismo do Neptune, a menos que ela seja usada dentro de uma etapa `sideEffect(` ou `optional()`.
- fold () — Existem apenas duas situações em que a etapa fold () pode ser convertida, a saber, quando é usada no padrão Upsert recomendado na página de TinkerPop receitas e quando é usada em um `group().by()` contexto como este:

```
g.V().has('code','ANC').out().group().by().by(values('code', 'city').fold())
```

- id( ): a etapa `id()` é convertida, a menos que seja usada em uma propriedade, como esta:

```
g.V().has('code','ANC').properties('code').id()
```

- order( ): em geral, a etapa `order()` pode ser convertida em uma operação nativa do mecanismo do Neptune, a menos que o seguinte seja verdadeiro:
  - A etapa `order()` está dentro de um percurso secundário aninhado, desta forma:

```
g.V().has('code','ANC').where(V().out().order().by(id))
```

  - A ordenação local está sendo usada, por exemplo, com `order(local)`.
  - Um comparador personalizado está sendo usado na modulação `by()` pela qual ordenar. Um exemplo é este uso de `sack()`:

```
g.withSack(0).
  V().has('code','ANC').
      repeat(outE().sack(sum).by('dist').inV()).times(2).limit(10).
      order().by(sack())
```

  - Há várias ordenações no mesmo elemento.
- project( ): em geral, a etapa `project()` pode ser convertida em uma operação nativa do mecanismo do Neptune, a menos que o número de declarações `by()` após a `project()` não corresponda ao número de rótulos especificado, como aqui:

```
g.V().has('code','ANC').project('x', 'y').by(id)
```

- range( ): a etapa `range()` só é convertida quando a extremidade inferior do intervalo em questão é zero (por exemplo, `range(0,3)`).
- repeat( ): em geral, a etapa `repeat()` pode ser convertida em uma operação nativa do mecanismo do Neptune, a menos que esteja aninhada em outra etapa `repeat()`, da seguinte forma:

```
g.V().has('code','ANC').repeat(out().repeat(out()).times(2)).times(2)
```

- sack( ): em geral, a etapa `sack()` pode ser convertida em uma operação nativa do mecanismo do Neptune, exceto nos seguintes casos:
  - Se um operador sack não numérico estiver sendo usado.
  - Se um operador sack numérico diferente de `+`, `-`, `mult`, `div`, `min` e `max` estiver sendo usado.
  - Se `sack()` for usado dentro de uma etapa `where()` para filtrar com base em um valor de sack, como aqui:

```
g.V().has('code','ANC').sack(assign).by(values('code')).where(sack().is('ANC'))
```

- sum( ): em geral, a etapa `sum()` pode ser convertida em uma operação nativa do mecanismo do Neptune, mas não quando usada para calcular uma soma global, como esta:

```
g.V().has('code','ANC').outE('routes').values('dist').sum()
```

- union( ): a etapa `union()` pode ser convertida em uma operação nativa do mecanismo do Neptune, desde que seja a última etapa da consulta, exceto a etapa do terminal.
- unfold () — A `unfold()` etapa só pode ser convertida em uma operação nativa do Neptune Engine quando usada no padrão Upsert recomendado na TinkerPop página de receitas e quando usada junto com esta: `cap()`

```
g.V().has('airport','country','IE').aggregate('airport').limit(2)
     .cap('airport').unfold()
```

- where( ): em geral, a etapa `where()` pode ser convertida em uma operação nativa do mecanismo do Neptune, exceto nos seguintes casos:
  - Quando as modulações by() são usadas, desta forma:

    ```
    g.V().hasLabel('airport').as('a')
         .where(gt('a')).by('runways')
    ```

  - Quando operadores de comparação diferentes de `eq`, `neq`, `within` e `without` são usados.
  - Quando agregações fornecidas pelo usuário são usadas.

Etapas do Gremlin que nunca são convertidas em operações nativas do mecanismo do Neptune

As seguintes etapas do Gremlin são compatíveis com o Neptune, mas nunca são convertidas em operações nativas do mecanismo do Neptune. Em vez disso, elas são executadas pelo servidor do Gremlin.

- choose( )
- coin( )
- inject( )
- match( )
- math( )
- max( )
- mean( )
- min( )
- option( )
- optional( )
- path( )
- propertyMap( )
- sample( )
- skip( )
- tail( )
- timeLimit( )
- tree( )

Etapas do Gremlin não compatíveis com o Neptune

As etapas do Gremlin a seguir não são de forma nenhuma compatíveis com o Neptune. Na maioria dos casos, o motivo disso é que elas exigem um `GraphComputer`, que não é aceito pelo Neptune no momento.

- connectedComponent( )
- io( )
- shortestPath( )
- withComputer( )
- pageRank( )
- peerPressure( )
- program( )

Na verdade, a etapa `io()` é parcialmente compatível, pois pode ser usada em `read()` partir de um URL, mas não em `write()`.

Usar o Gremlin com o mecanismo de consulta do DFE do Neptune

Se você habilitar totalmente o mecanismo de consulta alternativo do Neptune, conhecido como DFE, no modo de laboratório (definindo o parâmetro de cluster de banco de dados `neptune_lab_mode` como `DFEQueryEngine=enabled`), o Neptune converterá consultas/percursos do somente leitura Gremlin em uma representação lógica intermediária e os executará no mecanismo do DFE sempre que possível.

No entanto, o DFE ainda não é compatível com todas as etapas do Gremlin. Quando uma etapa não pode ser executada nativamente no DFE, Neptune volta a executá-la. TinkerPop Os relatórios `explain` e `profile` incluem avisos quando isso acontece.

> **Note**
>
> A partir da versão 1.0.5.0 do mecanismo, o comportamento padrão do DFE para lidar com etapas do Gremlin sem compatibilidade nativa mudou. Onde antes o motor DFE recaía sobre o motor Neptune Gremlin, agora ele volta para o motor vanilla. TinkerPop

Etapas do Gremlin que são compatíveis nativamente com o mecanismo do DFE

- **`GraphStep`**
- **`VertexStep`**
- **`EdgeVertexStep`**
- **`IdStep`**
- **`TraversalFilterStep`**
- **`PropertiesStep`**
- Suporte de filtragem **`HasStep`** para vértices e bordas em propriedades, IDs e rótulos, com exceção de texto e predicados `Without`.
- **`WherePredicateStep`** com filtros com escopo de `Path` definido, mas sem suporte a pesquisa `ByModulation`, `SideEffect` ou `Map`
- **`DedupGlobalStep`**, exceto suporte a pesquisa `ByModulation`, `SideEffect` e `Map`.

Intercalação do planejamento de consultas

Quando o processo de conversão encontra uma etapa do Gremlin sem um operador nativo do DFE correspondente, antes de voltar a usar o Tinkerpop, ele tenta encontrar outras partes intermediárias da consulta que possam ser executadas de modo nativo no mecanismo do DFE. Ele faz isso aplicando a lógica de intercalação ao percurso de nível superior. O resultado é que as etapas compatíveis são usadas sempre que possível.

Qualquer conversão de consulta intermediária, sem prefixo, é representada usando as saídas `NeptuneInterleavingStep`, `explain` e `profile`.

Para comparação de desempenho, convém desativar a intercalação em uma consulta e ainda usar o mecanismo do DFE para executar a parte do prefixo. Ou talvez você queira usar somente o TinkerPop mecanismo para execução de consultas sem prefixo. Você pode fazer isso usando a dica de consulta `disableInterleaving`.

Assim como a dica de consulta useDFE com um valor de `false` impede totalmente que uma consulta seja executada no DFE, a dica de consulta `disableInterleaving` com um valor de `true` desativa a intercalação do DFE para conversão de uma consulta. Por exemplo: .

```
g.with('Neptune#disableInterleaving', true)
 .V().has('genre','drama').in('likes')
```

Atualização da saída **`explain`** e **`profile`** do Gremlin

O explain do Gremlin fornece detalhes sobre o percurso otimizado que o Neptune usa para executar uma consulta. Consulte o exemplo de saída de `explain` do DFE para ver um exemplo de saída de `explain` quando o mecanismo do DFE está habilitado.

O API `profile` do Gremlin executa um percurso especificado do Gremlin, coleta várias métricas sobre a execução e produz um relatório de perfil que contém detalhes sobre o plano de consulta otimizado e as estatísticas de runtime de vários operadores. Consulte o exemplo de saída de `profile` do DFE para ver um exemplo de saída de `profile` quando o mecanismo do DFE está habilitado.

> **Note**
>
> O DFE é um atributo experimental lançado no modo de laboratório, portanto, o formato exato da saída `explain` e `profile` ainda está sujeito a alterações.

Acessar o grafo do Neptune com o openCypher

O Neptune é compatível com a criação de aplicações de grafos usando o openCypher, atualmente uma das linguagens de consulta mais populares para desenvolvedores que trabalham com bancos de dados de grafos. Desenvolvedores, analistas de negócios e cientistas de dados gostam da sintaxe inspirada em SQL do openCypher porque ela oferece uma estrutura familiar para compor consultas para aplicações de grafos.

openCypher é uma linguagem de consulta declarativa para grafos de propriedades originalmente desenvolvida pela Neo4j, que se tornou de código aberto em 2015, e contribuiu para o projeto openCypher sob uma licença de código aberto Apache 2. A sintaxe dela está documentada na Cypher Query Language Reference, versão 9.

Para conhecer as limitações e diferenças no suporte do Neptune à especificação do openCypher, consulte Conformidade com a especificação OpenCypher no Amazon Neptune.

> **Note**
>
> A implementação atual do Neo4j da linguagem de consulta Cypher divergiu em alguns aspectos da especificação do openCypher. Se você estiver migrando o código Neo4j Cypher

atual para o Neptune, consulte Compatibilidade do Neptune com o Neo4j e Reformular consultas do Cypher para serem executadas no openCypher no Neptune para obter ajuda.

A partir da versão 1.1.1.0 do mecanismo, o openCypher está disponível para uso em produção no Neptune.

Gremlin versus openCypher: semelhanças e diferenças

O Gremlin e o openCypher são linguagens de consulta de grafos de propriedades e são complementares de várias maneiras.

O Gremlin foi projetado para atrair programadores e se ajustar perfeitamente ao código. Como resultado, o Gremlin é fundamental por design, enquanto a sintaxe declarativa do openCypher pode parecer mais familiar para pessoas com experiência em SQL ou SPARQL. O Gremlin pode parecer mais natural para um cientista de dados que use Python em um caderno Jupyter, enquanto o openCypher pode parecer mais intuitivo para um usuário corporativo com alguma experiência em SQL.

O lado bom é que você não precisa escolher entre o Gremlin e o openCypher no Neptune. As consultas em qualquer linguagem podem funcionar no mesmo grafo, independentemente de qual das duas linguagens foi usada para inserir esses dados. Você pode achar mais conveniente usar o Gremlin para algumas coisas e o openCypher para outras, dependendo do que estiver fazendo.

O Gremlin usa uma sintaxe imperativa que permite controlar como você se move pelo grafo em uma série de etapas, cada uma das quais recebe um fluxo de dados, executa alguma ação nele (usando um filtro, um mapa, etc.) e, depois, envia os resultados para a próxima etapa. Uma consulta do Gremlin geralmente assume o formato `g.V()`, seguida por etapas adicionais.

No openCypher, você usa uma sintaxe declarativa, inspirada no SQL, que especifica um padrão de nós e relacionamentos a serem encontrados no grafo usando uma sintaxe de motivo (como `()-[]->()`). Uma consulta do openCypher geralmente começa com uma cláusula `MATCH`, seguida por outras cláusulas, como `WHERE`, `WITH` e `RETURN`.

Conceitos básicos sobre o openCypher

Você pode consultar dados do grafo de propriedades no Neptune usando o openCypher, independentemente de como eles foram carregados, mas não pode usar o openCypher para consultar dados carregados como RDF.

O carregador em massa do Neptune aceita dados de grafos de propriedades em formato CSV para Gremlin e em formato CSV para openCypher. Além disso, naturalmente, é possível adicionar dados de propriedades ao grafo usando consultas do Gremlin e/ou do openCypher.

Há muitos tutoriais on-line disponíveis para aprender a linguagem de consulta Cypher. Aqui, alguns exemplos rápidos de consultas do openCypher podem ajudar você a ter uma ideia da linguagem, mas, de longe, a maneira mais fácil e melhor de começar a usar o openCypher para consultar o grafo do Neptune é usando os cadernos openCypher na bancada de trabalho do Neptune. O ambiente de trabalho é de código aberto e está hospedado GitHub em https://github.com/aws-samples/amazon-neptune-samples.

Você encontrará os cadernos OpenCypher no repositório de cadernos gráficos GitHub Neptune. Especificamente, confira a Air-routes visualization e os cadernos de English Premier Teams para openCypher.

Os dados processados pelo openCypher assumem a forma de uma série não ordenada de mapas de chave/valor. A principal forma de refinar, manipular e incrementar esses mapas é usar cláusulas que executam tarefas como correspondência de padrões, inserção, atualização e exclusão nos pares chave/valor.

Há várias cláusulas no openCypher para encontrar padrões de dados no grafo, das quais `MATCH` é a mais comum. `MATCH` permite especificar o padrão de nós, relacionamentos e filtros que você deseja procurar no grafo. Por exemplo: .

- Obter todos os nós

```
MATCH (n) RETURN n
```

- Encontrar nós conectados

```
MATCH (n)-[r]->(d) RETURN n, r, d
```

- Encontrar um caminho

```
MATCH p=(n)-[r]->(d) RETURN p
```

- Obter todos os nós com um rótulo

```
MATCH (n:airport) RETURN n
```

Observe que a primeira consulta acima gera cada nó no grafo, e as duas seguintes geram cada nó que tem um relacionamento: isso geralmente não é recomendado! Em quase todos os casos, você deseja restringir os dados gerados, o que pode ser feito especificando rótulos e propriedades de nós ou relacionamentos, como no quarto exemplo.

É possível encontrar uma folha de dicas útil para a sintaxe do openCypher no repositório de exemplos do github do Neptune.

Endpoint de status e servlet de status do openCypher no Neptune

O endpoint de status do openCypher concede acesso a informações sobre consultas que estão sendo executadas no servidor ou aguardando execução. Também permite que você cancele essas consultas. O endpoint é:

```
https://(the server):(the port number)/openCypher/status
```

É possível usar os métodos `GET` e `POST` HTTP para obter o status atual do servidor ou cancelar uma consulta. Você também pode usar o método `DELETE` para cancelar uma consulta em execução ou em espera.

Parâmetros para solicitações de status

Parâmetros de consulta de status

- **`includeWaiting`** (`true` ou `false`): quando definido como `true` e outros parâmetros não estão presentes, faz com que as informações de status das consultas em espera sejam geradas, bem como das consultas em execução.
- **`cancelQuery`**: usado somente com os métodos `GET` e `POST`, para indicar que se trata de uma solicitação de cancelamento. O método `DELETE` não precisa desse parâmetro.

  O valor do parâmetro `cancelQuery` não é usado, mas quando `cancelQuery` está presente, o parâmetro `queryId` é obrigatório para identificar qual consulta cancelar.
- **`queryId`**: contém o ID de uma consulta específica.

  Quando usado com o método `GET` ou `POST` e o parâmetro `cancelQuery` não está presente, `queryId` faz com que as informações sejam geradas para a consulta específica identificada. Se o parâmetro `cancelQuery` estiver presente, a consulta específica que `queryId` identifica será cancelada.

Quando usado com o método `DELETE`, `queryId` sempre indica que uma consulta específica deve ser cancelada.

- **`silent`**: usado somente ao cancelar uma consulta. Se definido como `true`, faz com que o cancelamento ocorra silenciosamente.

Campos de resposta da solicitação de status

Campos de resposta de status se o ID de uma consulta específica não for fornecido

- aceito QueryCount — O número de consultas que foram aceitas, mas ainda não concluídas, incluindo consultas na fila.
- em execução QueryCount — O número de consultas do OpenCypher em execução no momento.
- queries: uma lista das consultas atuais do openCypher.

Campos de resposta de status para uma consulta específica

- queryId: um ID de GUID para a consulta. O Neptune atribui automaticamente esse valor de ID a cada consulta, ou você também pode atribuir seu próprio ID (consulte Injetar um ID personalizado em uma consulta do Gremlin ou do SPARQL no Neptune).
- queryString: a consulta enviada. Ela será truncada para 1024 caracteres se for maior do que isso.
- consulta EvalStats — Estatísticas dessa consulta:
  - waited: indica quanto tempo a consulta esperou, em milissegundos.
  - elapsed: o número de milissegundos em que a consulta esteve em execução até o momento.
  - cancelled: `True` indica que a consulta foi cancelada ou `False` que não foi cancelada.

Exemplos de solicitação e resposta de status

- Solicitação do status de todas as consultas, incluindo aquelas em espera:

```
curl https://server:port/openCypher/status \
  --data-urlencode "includeWaiting=true"
```

Resposta:

```
{
```

```
  "acceptedQueryCount" : 0,
  "runningQueryCount" : 0,
  "queries" : [ ]
}
```

- Solicitação do status de todas as consultas em execução, não incluindo aquelas em espera:

```
curl https://server:port/openCypher/status
```

Resposta:

```
{
  "acceptedQueryCount" : 0,
  "runningQueryCount" : 0,
  "queries" : [ ]
}
```

- Solicitação do status de uma única consulta:

```
curl https://server:port/openCypher/status \
 --data-urlencode "queryId=eadc6eea-698b-4a2f-8554-5270ab17ebee"
```

Resposta:

```
{
  "queryId" : "eadc6eea-698b-4a2f-8554-5270ab17ebee",
  "queryString" : "MATCH (n1)-[:knows]->(n2), (n2)-[:knows]->(n3), (n3)-[:knows]-
>(n4), (n4)-[:knows]->(n5), (n5)-[:knows]->(n6), (n6)-[:knows]->(n7), (n7)-[:knows]-
>(n8), (n8)-[:knows]->(n9), (n9)-[:knows]->(n10) RETURN COUNT(n1);",
  "queryEvalStats" : {
    "waited" : 0,
    "elapsed" : 23463,
    "cancelled" : false
  }
}
```

- Solicitações para cancelar uma consulta

1. Usar POST:

```
curl -X POST https://server:port/openCypher/status \
  --data-urlencode "cancelQuery" \
```

```
  --data-urlencode "queryId=f43ce17b-db01-4d37-a074-c76d1c26d7a9"
```

Resposta:

```
{
  "status" : "200 OK",
  "payload" : true
}
```

2. Usar GET:

```
curl -X GET https://server:port/openCypher/status \
  --data-urlencode "cancelQuery" \
  --data-urlencode "queryId=588af350-cfde-4222-bee6-b9cedc87180d"
```

Resposta:

```
{
  "status" : "200 OK",
  "payload" : true
}
```

3. Usar DELETE:

```
curl -X DELETE \
  -s "https://server:port/openCypher/status?queryId=b9a516d1-d25c-4301-
bb80-10b2743ecf0e"
```

Resposta:

```
{
  "status" : "200 OK",
  "payload" : true
}
```

O endpoint HTTPS do openCypher no Amazon Neptune

Tópicos

- Consultas de leitura e gravação do openCypher no endpoint HTTPS
- O formato padrão de resultados JSON do openCypher

Consultas de leitura e gravação do openCypher no endpoint HTTPS

O endpoint HTTPS do openCypher é compatível com consultas de leitura e atualização usando `GET` e o método `POST`. Os métodos `DELETE` e `PUT` não são compatíveis.

As instruções a seguir explicam como conectar-se ao endpoint do openCypher usando o comando `curl` e HTTPS. Você deve seguir estas instruções em uma instância do Amazon EC2 na mesma nuvem privada virtual (VPC) que a instância de banco de dados do Neptune.

A sintaxe é:

```
HTTPS://(the server):(the port number)/openCypher
```

Veja exemplos de consultas de leitura, uma que usa `POST` e outra que usa `GET`:

1. Usar `POST`:

```
curl HTTPS://server:port/openCypher \
  -d "query=MATCH (n1) RETURN n1;"
```

2. Usar `GET` (a string de consulta é codificada em URL):

```
curl -X GET \
  "HTTPS://server:port/openCypher?query=MATCH%20(n1)%20RETURN%20n1"
```

Veja exemplos de consulta de gravação/atualização, uma que usa `POST` e outra que usa `GET`:

1. Usar `POST`:

```
curl HTTPS://server:port/openCypher \
  -d "query=CREATE (n:Person { age: 25 })"
```

2. Usar `GET` (a string de consulta é codificada em URL):

```
curl -X GET \
  "HTTPS://server:port/openCypher?query=CREATE%20(n%3APerson%20%7B%20age%3A
%2025%20%7D)"
```

O formato padrão de resultados JSON do openCypher

O formato JSON a seguir é gerado por padrão ou definindo explicitamente o cabeçalho de solicitação como `Accept: application/json`. Esse formato foi projetado para ser facilmente analisado em objetos usando atributos de linguagem nativa da maioria das bibliotecas.

O documento JSON gerado apresenta um campo, `results`, que contém os valores de retorno da consulta. Os exemplos abaixo mostram a formatação JSON para valores comuns.

Exemplo de resposta de valor:

```
{
  "results": [
    {
      "count(a)": 121
    }
  ]
}
```

Exemplo de resposta do nó:

```
{
  "results": [
    {
      "a": {
        "~id": "22",
        "~entityType": "node",
        "~labels": [
          "airport"
        ],
        "~properties": {
          "desc": "Seattle-Tacoma",
          "lon": -122.30899810791,
          "runways": 3,
          "type": "airport",
          "country": "US",
          "region": "US-WA",
          "lat": 47.4490013122559,
          "elev": 432,
          "city": "Seattle",
          "icao": "KSEA",
          "code": "SEA",
          "longest": 11901
```

```
        }
      }
    }
  ]
}
```

Exemplo de resposta de relacionamento:

```
{
  "results": [
    {
      "r": {
        "~id": "7389",
        "~entityType": "relationship",
        "~start": "22",
        "~end": "151",
        "~type": "route",
        "~properties": {
          "dist": 956
        }
      }
    }
  ]
}
```

Exemplo de resposta de caminho:

```
{
  "results": [
    {
      "p": [
        {
          "~id": "22",
          "~entityType": "node",
          "~labels": [
            "airport"
          ],
          "~properties": {
            "desc": "Seattle-Tacoma",
            "lon": -122.30899810791,
            "runways": 3,
            "type": "airport",
            "country": "US",
```

```
            "region": "US-WA",
            "lat": 47.4490013122559,
            "elev": 432,
            "city": "Seattle",
            "icao": "KSEA",
            "code": "SEA",
            "longest": 11901
          }
        },
        {
          "~id": "7389",
          "~entityType": "relationship",
          "~start": "22",
          "~end": "151",
          "~type": "route",
          "~properties": {
            "dist": 956
          }
        },
        {
          "~id": "151",
          "~entityType": "node",
          "~labels": [
            "airport"
          ],
          "~properties": {
            "desc": "Ontario International Airport",
            "lon": -117.600997924805,
            "runways": 2,
            "type": "airport",
            "country": "US",
            "region": "US-CA",
            "lat": 34.0559997558594,
            "elev": 944,
            "city": "Ontario",
            "icao": "KONT",
            "code": "ONT",
            "longest": 12198
          }
        }
      ]
    }
  ]
```

```
}
```

Usar o protocolo Bolt para fazer consultas do openCypher ao Neptune

O Bolt é um protocolo cliente/servidor orientado a declarações inicialmente desenvolvido pela Neo4j e licenciado sob a licença Creative Commons 3.0 Attribution-. ShareAlike É orientado por cliente, o que significa que o cliente sempre inicia as trocas de mensagens.

Para se conectar ao Neptune usando os drivers Bolt da Neo4j, basta substituir o URL e o número da porta pelos endpoints de cluster usando o esquema URI `bolt`. Se você tiver uma única instância do Neptune em execução, use o endpoint read_write. Caso várias instâncias estejam em execução, dois drivers são recomendados, um para o gravador e outro para todas as réplicas de leitura. Caso você tenha apenas dois endpoints padrão, um driver read_write e um read_only são suficientes, mas se também tiver endpoints personalizados, pense em criar uma instância de driver para cada um.

> **Note**
>
> Embora a especificação do Bolt afirme que o Bolt pode se conectar usando TCP ou WebSockets, o Neptune suporta apenas conexões TCP para o Bolt.

O Neptune permite até mil conexões Bolt simultâneas.

Para obter exemplos de consulta do openCypher em várias linguagens que usem os drivers Bolt, consulte a documentação Drivers & Language Guides da Neo4j.

> **Important**
>
> Os drivers Neo4j Bolt para Python JavaScript, .NET e Golang inicialmente não suportavam a renovação automática dos tokens de autenticação Signature v4. AWS Isso significa que, após a expiração da assinatura (geralmente em cinco minutos), o driver não foi autenticado e ocorreu uma falha nas solicitações subsequentes. Os exemplos de Python JavaScript, .NET e Go abaixo foram todos afetados por esse problema.
> Consulte o problema #834 do driver Neo4j Python, o problema#664 do Neo4j .NET, o problema #993 do driver Neo4j e o problema #429 do JavaScript driver Neo4j GoLang para obter mais informações.
> A partir da versão 5.8.0 do driver, uma nova versão prévia da API de reautenticação foi lançada para o driver Go (consulte v5.8.0 - Feedback wanted on re-authentication).

Usar o Bolt com Java para conectar-se ao Neptune

É possível baixar um driver para qualquer versão que quiser usar do MVN Repository do Maven ou adicionar essa dependência ao projeto:

```
<dependency>
  <groupId>org.neo4j.driver</groupId>
  <artifactId>neo4j-java-driver</artifactId>
  <version>4.3.3</version>
</dependency>
```

Depois, para se conectar ao Neptune em Java usando um desses drivers Bolt, crie uma instância de driver para a instância primária/de gravador no cluster usando um código como o seguinte:

```
import org.neo4j.driver.Driver;
import org.neo4j.driver.GraphDatabase;

final Driver driver =
  GraphDatabase.driver("bolt://(your cluster endpoint URL):(your cluster port)",
    AuthTokens.none(),
    Config.builder().withEncryption()
                    .withTrustStrategy(TrustStrategy.trustSystemCertificates())
                    .build());
```

Se você tiver uma ou mais réplicas de leitor, também poderá criar uma instância de driver para elas usando um código como este:

```
final Driver read_only_driver =              // (without connection timeout)
  GraphDatabase.driver("bolt://(your cluster endpoint URL):(your cluster port)",
      Config.builder().withEncryption()
                      .withTrustStrategy(TrustStrategy.trustSystemCertificates())
                      .build());
```

Ou com um tempo limite:

```
final Driver read_only_timeout_driver =      // (with connection timeout)
  GraphDatabase.driver("bolt://(your cluster endpoint URL):(your cluster port)",
    Config.builder().withConnectionTimeout(30, TimeUnit.SECONDS)
                    .withEncryption()
                    .withTrustStrategy(TrustStrategy.trustSystemCertificates())
                    .build());
```

Se você tiver endpoints personalizados, também poderá valer a pena criar uma instância de driver para cada um.

Um exemplo de consulta do openCypher em Python usando Bolt

Veja como fazer uma consulta do openCypher em Python usando o Bolt:

```
python -m pip install neo4j
```

```
from neo4j import GraphDatabase
uri = "bolt://(your cluster endpoint URL):(your cluster port)"
driver = GraphDatabase.driver(uri, auth=("username", "password"), encrypted=True)
```

Observe que os parâmetros `auth` são ignorados.

Um exemplo de consulta do openCypher em .NET usando o Bolt

Para fazer uma consulta OpenCypher no.NET usando o Bolt, a primeira etapa é instalar o driver Neo4j usando. NuHet Para fazer chamadas síncronas, use a versão `.Simple`, desta forma:

```
Install-Package Neo4j.Driver.Simple-4.3.0
```

```
using Neo4j.Driver;

namespace hello
{
  // This example creates a node and reads a node in a Neptune
  // Cluster where IAM Authentication is not enabled.
  public class HelloWorldExample : IDisposable
  {
    private bool _disposed = false;
    private readonly IDriver _driver;
    private static string url = "bolt://(your cluster endpoint URL):(your cluster
 port)";
    private static string createNodeQuery = "CREATE (a:Greeting) SET a.message =
 'HelloWorldExample'";
    private static string readNodeQuery = "MATCH(n:Greeting) RETURN n.message";

    ~HelloWorldExample() => Dispose(false);

    public HelloWorldExample(string uri)
    {
```

```
      _driver = GraphDatabase.Driver(uri, AuthTokens.None, o =>
o.WithEncryptionLevel(EncryptionLevel.Encrypted));
   }

   public void createNode()
   {
     // Open a session
     using (var session = _driver.Session())
     {
        // Run the query in a write transaction
       var greeting = session.WriteTransaction(tx =>
       {
         var result = tx.Run(createNodeQuery);
         // Consume the result
         return result.Consume();
       });

       // The output will look like this:
       //   ResultSummary{Query=`CREATE (a:Greeting) SET a.message =
'HelloWorldExample".....
       Console.WriteLine(greeting);
     }
   }

   public void retrieveNode()
   {
     // Open a session
     using (var session = _driver.Session())
     {
       // Run the query in a read transaction
       var greeting = session.ReadTransaction(tx =>
       {
         var result = tx.Run(readNodeQuery);
         // Consume the result. Read the single node
         // created in a previous step.
         return result.Single()[0].As<string>();
       });
       // The output will look like this:
       //   HelloWorldExample
       Console.WriteLine(greeting);
     }
   }

   public void Dispose()
```

```
    {
      Dispose(true);
      GC.SuppressFinalize(this);
    }

    protected virtual void Dispose(bool disposing)
    {
      if (_disposed)
        return;
      if (disposing)
      {
        _driver?.Dispose();
      }
      _disposed = true;
    }

    public static void Main()
    {
      using (var apiCaller = new HelloWorldExample(url))
      {
        apiCaller.createNode();
        apiCaller.retrieveNode();
      }
    }
  }
}
```

Um exemplo de consulta do openCypher em Java usando o Bolt com autenticação do IAM

O código Java abaixo mostra como fazer consultas do openCypher em Java usando o Bolt com autenticação do IAM. O JavaDoc comentário descreve seu uso. Quando uma instância de driver estiver disponível, você poderá usá-la para fazer várias solicitações autenticadas.

```
package software.amazon.neptune.bolt;

import com.amazonaws.DefaultRequest;
import com.amazonaws.Request;
import com.amazonaws.auth.AWS4Signer;
import com.amazonaws.auth.AWSCredentialsProvider;
import com.amazonaws.http.HttpMethodName;
import com.google.gson.Gson;
import lombok.Builder;
```

```
import lombok.Getter;
import lombok.NonNull;
import org.neo4j.driver.Value;
import org.neo4j.driver.Values;
import org.neo4j.driver.internal.security.InternalAuthToken;
import org.neo4j.driver.internal.value.StringValue;

import java.net.URI;
import java.util.Collections;
import java.util.HashMap;
import java.util.Map;

import static com.amazonaws.auth.internal.SignerConstants.AUTHORIZATION;
import static com.amazonaws.auth.internal.SignerConstants.HOST;
import static com.amazonaws.auth.internal.SignerConstants.X_AMZ_DATE;
import static com.amazonaws.auth.internal.SignerConstants.X_AMZ_SECURITY_TOKEN;

/**
 * Use this class instead of `AuthTokens.basic` when working with an IAM
 * auth-enabled server. It works the same as `AuthTokens.basic` when using
 * static credentials, and avoids making requests with an expired signature
 * when using temporary credentials. Internally, it generates a new signature
 * on every invocation (this may change in a future implementation).
 *
 * Note that authentication happens only the first time for a pooled connection.
 *
 * Typical usage:
 *
 * NeptuneAuthToken authToken = NeptuneAuthToken.builder()
 *     .credentialsProvider(credentialsProvider)
 *     .region("aws region")
 *     .url("cluster endpoint url")
 *     .build();
 *
 * Driver driver = GraphDatabase.driver(
 *     authToken.getUrl(),
 *     authToken,
 *     config
 * );
 */

public class NeptuneAuthToken extends InternalAuthToken {
  private static final String SCHEME = "basic";
  private static final String REALM = "realm";
```

```
  private static final String SERVICE_NAME = "neptune-db";
  private static final String HTTP_METHOD_HDR = "HttpMethod";
  private static final String DUMMY_USERNAME = "username";
  @NonNull
  private final String region;
  @NonNull
  @Getter
  private final String url;
  @NonNull
  private final AWSCredentialsProvider credentialsProvider;
  private final Gson gson = new Gson();

  @Builder
  private NeptuneAuthToken(
      @NonNull final String region,
      @NonNull final String url,
      @NonNull final AWSCredentialsProvider credentialsProvider
  ) {
      // The superclass caches the result of toMap(), which we don't want
      super(Collections.emptyMap());
      this.region = region;
      this.url = url;
      this.credentialsProvider = credentialsProvider;
  }

  @Override
  public Map<String, Value> toMap() {
    final Map<String, Value> map = new HashMap<>();
    map.put(SCHEME_KEY, Values.value(SCHEME));
    map.put(PRINCIPAL_KEY, Values.value(DUMMY_USERNAME));
    map.put(CREDENTIALS_KEY, new StringValue(getSignedHeader()));
    map.put(REALM_KEY, Values.value(REALM));

    return map;
  }

  private String getSignedHeader() {
    final Request<Void> request = new DefaultRequest<>(SERVICE_NAME);
    request.setHttpMethod(HttpMethodName.GET);
    request.setEndpoint(URI.create(url));
    // Comment out the following line if you're using an engine version older than
1.2.0.0
    request.setResourcePath("/opencypher");
```

```
    final AWS4Signer signer = new AWS4Signer();
    signer.setRegionName(region);
    signer.setServiceName(request.getServiceName());
    signer.sign(request, credentialsProvider.getCredentials());

    return getAuthInfoJson(request);
  }

  private String getAuthInfoJson(final Request<Void> request) {
    final Map<String, Object> obj = new HashMap<>();
    obj.put(AUTHORIZATION, request.getHeaders().get(AUTHORIZATION));
    obj.put(HTTP_METHOD_HDR, request.getHttpMethod());
    obj.put(X_AMZ_DATE, request.getHeaders().get(X_AMZ_DATE));
    obj.put(HOST, request.getHeaders().get(HOST));
    obj.put(X_AMZ_SECURITY_TOKEN, request.getHeaders().get(X_AMZ_SECURITY_TOKEN));

    return gson.toJson(obj);
  }
}
```

Um exemplo de consulta do openCypher em Python usando o Bolt com autenticação do IAM

A classe Python abaixo permite que você faça consultas do openCypher em Python usando o Bolt com autenticação do IAM:

```
import json

from neo4j import Auth
from botocore.awsrequest import AWSRequest
from botocore.credentials import Credentials
from botocore.auth import (
  SigV4Auth,
  _host_from_url,
)

SCHEME = "basic"
REALM = "realm"
SERVICE_NAME = "neptune-db"
DUMMY_USERNAME = "username"
HTTP_METHOD_HDR = "HttpMethod"
HTTP_METHOD = "GET"
AUTHORIZATION = "Authorization"
```

```
X_AMZ_DATE = "X-Amz-Date"
X_AMZ_SECURITY_TOKEN = "X-Amz-Security-Token"
HOST = "Host"


class NeptuneAuthToken(Auth):
  def __init__(
    self,
    credentials: Credentials,
    region: str,
    url: str,
    **parameters
  ):
    # Do NOT add "/opencypher" in the line below if you're using an engine version
 older than 1.2.0.0
    request = AWSRequest(method=HTTP_METHOD, url=url + "/opencypher")
    request.headers.add_header("Host", _host_from_url(request.url))
    sigv4 = SigV4Auth(credentials, SERVICE_NAME, region)
    sigv4.add_auth(request)

    auth_obj = {
      hdr: request.headers[hdr]
      for hdr in [AUTHORIZATION, X_AMZ_DATE, X_AMZ_SECURITY_TOKEN, HOST]
    }
    auth_obj[HTTP_METHOD_HDR] = request.method
    creds: str = json.dumps(auth_obj)
    super().__init__(SCHEME, DUMMY_USERNAME, creds, REALM, **parameters)
```

Você usa essa classe para criar um driver da seguinte forma:

```
  authToken = NeptuneAuthToken(creds, REGION, URL)
  driver = GraphDatabase.driver(URL, auth=authToken, encrypted=True)
```

Um exemplo de Node.js usando a autenticação do IAM e o Bolt

O código Node.js abaixo usa a sintaxe do AWS SDK para a JavaScript versão 3 e ES6 para criar um driver que autentica as solicitações:

```
import neo4j from "neo4j-driver";
import { HttpRequest }  from "@aws-sdk/protocol-http";
import { defaultProvider } from "@aws-sdk/credential-provider-node";
import { SignatureV4 } from "@aws-sdk/signature-v4";
import crypto from "@aws-crypto/sha256-js";
```

```
const { Sha256 } = crypto;
import assert from "node:assert";

const region = "us-west-2";
const serviceName = "neptune-db";
const host = "(your cluster endpoint URL)";
const port = 8182;
const protocol = "bolt";
const hostPort = host + ":" + port;
const url = protocol + "://" + hostPort;
const createQuery = "CREATE (n:Greeting {message: 'Hello'}) RETURN ID(n)";
const readQuery = "MATCH(n:Greeting) WHERE ID(n) = $id RETURN n.message";

async function signedHeader() {
  const req = new HttpRequest({
    method: "GET",
    protocol: protocol,
    hostname: host,
    port: port,
    // Comment out the following line if you're using an engine version older than
 1.2.0.0
    path: "/opencypher",
    headers: {
      host: hostPort
    }
  });

  const signer = new SignatureV4({
    credentials: defaultProvider(),
    region: region,
    service: serviceName,
    sha256: Sha256
  });

  return signer.sign(req, { unsignableHeaders: new Set(["x-amz-content-sha256"]) })
    .then((signedRequest) => {
      const authInfo = {
        "Authorization": signedRequest.headers["authorization"],
        "HttpMethod": signedRequest.method,
        "X-Amz-Date": signedRequest.headers["x-amz-date"],
        "Host": signedRequest.headers["host"],
        "X-Amz-Security-Token": signedRequest.headers["x-amz-security-token"]
      };
      return JSON.stringify(authInfo);
```

```
    });
}

async function createDriver() {
  let authToken = { scheme: "basic", realm: "realm", principal: "username",
 credentials: await signedHeader() };

  return neo4j.driver(url, authToken, {
      encrypted: "ENCRYPTION_ON",
      trust: "TRUST_SYSTEM_CA_SIGNED_CERTIFICATES",
      maxConnectionPoolSize: 1,
      // logging: neo4j.logging.console("debug")
    }
  );
}

function unmanagedTxn(driver) {
  const session = driver.session();
  const tx = session.beginTransaction();
  tx.run(createQuery)
  .then((res) => {
    const id = res.records[0].get(0);
    return tx.run(readQuery, { id: id });
  })
  .then((res) => {
    // All good, the transaction will be committed
    const msg = res.records[0].get("n.message");
    assert.equal(msg, "Hello");
  })
  .catch(err => {
    // The transaction will be rolled back, now handle the error.
    console.log(err);
  })
  .then(() => session.close());
}

createDriver()
.then((driver) => {
  unmanagedTxn(driver);
  driver.close();
})
.catch((err) => {
  console.log(err);
```

```
});
```

Um exemplo de consulta do openCypher em .NET usando o Bolt com autenticação do IAM

Para habilitar a autenticação do IAM em .NET, é necessário assinar uma solicitação ao estabelecer a conexão. O exemplo abaixo mostra como criar um auxiliar `NeptuneAuthToken` para gerar um token de autenticação:

```
using Amazon.Runtime;
using Amazon.Util;
using Neo4j.Driver;
using System.Security.Cryptography;
using System.Text;
using System.Text.Json;
using System.Web;

namespace Hello
{
  /*
   * Use this class instead of `AuthTokens.None` when working with an IAM-auth-enabled
 server.
   *
   * Note that authentication happens only the first time for a pooled connection.
   *
   * Typical usage:
   *
   * var authToken = new NeptuneAuthToken(AccessKey, SecretKey,
 Region).GetAuthToken(Host);
   * _driver = GraphDatabase.Driver(Url, authToken, o =>
 o.WithEncryptionLevel(EncryptionLevel.Encrypted));
   */

  public class NeptuneAuthToken
  {
    private const string ServiceName = "neptune-db";
    private const string Scheme = "basic";
    private const string Realm = "realm";
    private const string DummyUserName = "username";
    private const string Algorithm = "AWS4-HMAC-SHA256";
    private const string AWSRequest = "aws4_request";

    private readonly string _accessKey;
```

```
    private readonly string _secretKey;
    private readonly string _region;

    private readonly string _emptyPayloadHash;

    private readonly SHA256 _sha256;


    public NeptuneAuthToken(string awsKey = null, string secretKey = null, string
region = null)
    {
      var awsCredentials = awsKey == null || secretKey == null
        ? FallbackCredentialsFactory.GetCredentials().GetCredentials()
        : null;

      _accessKey = awsKey ?? awsCredentials.AccessKey;
      _secretKey = secretKey ?? awsCredentials.SecretKey;
      _region = region ?? FallbackRegionFactory.GetRegionEndpoint().SystemName; //ex:
us-east-1

      _sha256 = SHA256.Create();
      _emptyPayloadHash = Hash(Array.Empty<byte>());
    }

    public IAuthToken GetAuthToken(string url)
    {
      return AuthTokens.Custom(DummyUserName, GetCredentials(url), Realm, Scheme);
    }

    /******************* AWS SIGNING FUNCTIONS ********************/
    private string Hash(byte[] bytesToHash)
    {
      return ToHexString(_sha256.ComputeHash(bytesToHash));
    }

    private static byte[] HmacSHA256(byte[] key, string data)
    {
      return new HMACSHA256(key).ComputeHash(Encoding.UTF8.GetBytes(data));
    }

    private byte[] GetSignatureKey(string dateStamp)
    {
      var kSecret = Encoding.UTF8.GetBytes($"AWS4{_secretKey}");
      var kDate = HmacSHA256(kSecret, dateStamp);
```

```
      var kRegion = HmacSHA256(kDate, _region);
      var kService = HmacSHA256(kRegion, ServiceName);
      return HmacSHA256(kService, AWSRequest);
    }

    private static string ToHexString(byte[] array)
    {
      return Convert.ToHexString(array).ToLowerInvariant();
    }

    private string GetCredentials(string url)
    {
      var request = new HttpRequestMessage
      {
        Method = HttpMethod.Get,
        RequestUri = new Uri($"https://{url}/opencypher")
      };

      var signedrequest = Sign(request);

      var headers = new Dictionary<string, object>
      {
        [HeaderKeys.AuthorizationHeader] =
signedrequest.Headers.GetValues(HeaderKeys.AuthorizationHeader).FirstOrDefault(),
        ["HttpMethod"] = HttpMethod.Get.ToString(),
        [HeaderKeys.XAmzDateHeader] =
signedrequest.Headers.GetValues(HeaderKeys.XAmzDateHeader).FirstOrDefault(),
        // Host should be capitalized, not like in Amazon.Util.HeaderKeys.HostHeader
        ["Host"] =
signedrequest.Headers.GetValues(HeaderKeys.HostHeader).FirstOrDefault(),
      };

      return JsonSerializer.Serialize(headers);
    }

    private HttpRequestMessage Sign(HttpRequestMessage request)
    {
      var now = DateTimeOffset.UtcNow;
      var amzdate = now.ToString("yyyyMMddTHHmmssZ");
      var datestamp = now.ToString("yyyyMMdd");

      if (request.Headers.Host == null)
      {
        request.Headers.Host = $"{request.RequestUri.Host}:{request.RequestUri.Port}";
```

```
      }

      request.Headers.Add(HeaderKeys.XAmzDateHeader, amzdate);

      var canonicalQueryParams = GetCanonicalQueryParams(request);

      var canonicalRequest = new StringBuilder();
      canonicalRequest.Append(request.Method + "\n");
      canonicalRequest.Append(request.RequestUri.AbsolutePath + "\n");
      canonicalRequest.Append(canonicalQueryParams + "\n");

      var signedHeadersList = new List<string>();
      foreach (var header in request.Headers.OrderBy(a => a.Key.ToLowerInvariant()))
      {
        canonicalRequest.Append(header.Key.ToLowerInvariant());
        canonicalRequest.Append(':');
        canonicalRequest.Append(string.Join(",", header.Value.Select(s => s.Trim())));
        canonicalRequest.Append('\n');
        signedHeadersList.Add(header.Key.ToLowerInvariant());
      }
      canonicalRequest.Append('\n');

      var signedHeaders = string.Join(";", signedHeadersList);
      canonicalRequest.Append(signedHeaders + "\n");
      canonicalRequest.Append(_emptyPayloadHash);

      var credentialScope = $"{datestamp}/{_region}/{ServiceName}/{AWSRequest}";
      var stringToSign = $"{Algorithm}\n{amzdate}\n{credentialScope}\n"
        + Hash(Encoding.UTF8.GetBytes(canonicalRequest.ToString()));

      var signing_key = GetSignatureKey(datestamp);
      var signature = ToHexString(HmacSHA256(signing_key, stringToSign));

      request.Headers.TryAddWithoutValidation(HeaderKeys.AuthorizationHeader,
        $"{Algorithm} Credential={_accessKey}/{credentialScope},
SignedHeaders={signedHeaders}, Signature={signature}");

      return request;
    }

    private static string GetCanonicalQueryParams(HttpRequestMessage request)
    {
      var querystring = HttpUtility.ParseQueryString(request.RequestUri.Query);
```

```
      // Query params must be escaped in upper case (i.e. "%2C", not "%2c").
      var queryParams = querystring.AllKeys.OrderBy(a => a)
        .Select(key => $"{key}={Uri.EscapeDataString(querystring[key])}");
      return string.Join("&", queryParams);
    }
  }
}
```

Veja como fazer uma consulta do openCypher em .NET usando o Bolt com autenticação do IAM. O exemplo abaixo usa o auxiliar `NeptuneAuthToken`:

```
using Neo4j.Driver;

namespace Hello
{
  public class HelloWorldExample
  {
    private const string Host = "(your hostname):8182";
    private const string Url = $"bolt://{Host}";
    private const string CreateNodeQuery = "CREATE (a:Greeting) SET a.message =
 'HelloWorldExample'";
    private const string ReadNodeQuery = "MATCH(n:Greeting) RETURN n.message";

    private const string AccessKey = "(your access key)";
    private const string SecretKey = "(your secret key)";
    private const string Region = "(your AWS region)"; // e.g. "us-west-2"

    private readonly IDriver _driver;

    public HelloWorldExample()
    {
      var authToken = new NeptuneAuthToken(AccessKey, SecretKey,
 Region).GetAuthToken(Host);

      // Note that when the connection is reinitialized after max connection lifetime
      // has been reached, the signature token could have already been expired (usually
 5 min)
      // You can face exceptions like:
      //   `Unexpected server exception 'Signature expired: XXXX is now earlier than
 YYYY (ZZZZ - 5 min.)`
      _driver = GraphDatabase.Driver(Url, authToken, o =>

 o.WithMaxConnectionLifetime(TimeSpan.FromMinutes(60)).WithEncryptionLevel(EncryptionLevel.Encr
```

```
    }

    public async Task CreateNode()
    {
      // Open a session
      using (var session = _driver.AsyncSession())
      {
        // Run the query in a write transaction
        var greeting = await session.WriteTransactionAsync(async tx =>
        {
          var result = await tx.RunAsync(CreateNodeQuery);
          // Consume the result
          return await result.ConsumeAsync();
        });

        // The output will look like this:
        //   ResultSummary{Query=`CREATE (a:Greeting) SET a.message =
'HelloWorldExample".....
        Console.WriteLine(greeting.Query);
      }
    }

    public async Task RetrieveNode()
    {
      // Open a session
      using (var session = _driver.AsyncSession())
      {
        // Run the query in a read transaction
        var greeting = await session.ReadTransactionAsync(async tx =>
        {
          var result = await tx.RunAsync(ReadNodeQuery);
          var records = await result.ToListAsync();

          // Consume the result. Read the single node
          // created in a previous step.
          return records[0].Values.First().Value;
        });
        // The output will look like this:
        //   HelloWorldExample
        Console.WriteLine(greeting);
      }
    }
  }
```

```
}
```

Esse exemplo pode ser iniciado executando o código abaixo em `.NET 6` ou `.NET 7` com os seguintes pacotes:

- **`Neo4j`**`.Driver=4.3.0`
- **`AWSSDK`**`.Core=3.7.102.1`

```
namespace Hello
{
  class Program
  {
    static async Task Main()
    {
      var apiCaller = new HelloWorldExample();

      await apiCaller.CreateNode();
      await apiCaller.RetrieveNode();
    }
  }
}
```

Um exemplo de consulta do openCypher em Golang usando o Bolt com autenticação do IAM

O pacote Golang abaixo mostra como fazer consultas do openCypher na linguagem Go usando o Bolt com autenticação do IAM:

```
package main

import (
  "context"
  "encoding/json"
  "fmt"
  "github.com/aws/aws-sdk-go/aws/credentials"
  "github.com/aws/aws-sdk-go/aws/signer/v4"
  "github.com/neo4j/neo4j-go-driver/v5/neo4j"
  "log"
  "net/http"
  "os"
  "time"
```

```
)

const (
  ServiceName   = "neptune-db"
  DummyUsername = "username"
)

// Find node by id using Go driver
func findNode(ctx context.Context, region string, hostAndPort string, nodeId string)
 (string, error) {
  req, err := http.NewRequest(http.MethodGet, "https://"+hostAndPort+"/opencypher",
 nil)

  if err != nil {
    return "", fmt.Errorf("error creating request, %v", err)
  }

  // credentials must have been exported as environment variables
  signer := v4.NewSigner(credentials.NewEnvCredentials())
  _, err = signer.Sign(req, nil, ServiceName, region, time.Now())

  if err != nil {
    return "", fmt.Errorf("error signing request: %v", err)
  }

  hdrs := []string{"Authorization", "X-Amz-Date", "X-Amz-Security-Token"}
  hdrMap := make(map[string]string)
  for _, h := range hdrs {
    hdrMap[h] = req.Header.Get(h)
  }

  hdrMap["Host"] = req.Host
  hdrMap["HttpMethod"] = req.Method

  password, err := json.Marshal(hdrMap)
  if err != nil {
    return "", fmt.Errorf("error creating JSON, %v", err)
  }
  authToken := neo4j.BasicAuth(DummyUsername, string(password), "")
  // +s enables encryption with a full certificate check
  // Use +ssc to disable client side TLS verification
  driver, err := neo4j.NewDriverWithContext("bolt+s://"+hostAndPort+"/opencypher",
 authToken)
  if err != nil {
```

```
    return "", fmt.Errorf("error creating driver, %v", err)
  }

  defer driver.Close(ctx)

  if err := driver.VerifyConnectivity(ctx); err != nil {
    log.Fatalf("failed to verify connection, %v", err)
  }

  config := neo4j.SessionConfig{}

  session := driver.NewSession(ctx, config)
  defer session.Close(ctx)

  result, err := session.Run(
    ctx,
    fmt.Sprintf("MATCH (n) WHERE ID(n) = '%s' RETURN n", nodeId),
    map[string]any{},
  )
  if err != nil {
    return "", fmt.Errorf("error running query, %v", err)
  }

  if !result.Next(ctx) {
    return "", fmt.Errorf("node not found")
  }

  n, found := result.Record().Get("n")
  if !found {
    return "", fmt.Errorf("node not found")
  }

  return fmt.Sprintf("+%v\n", n), nil
}

func main() {
  if len(os.Args) < 3 {
    log.Fatal("Usage: go main.go (region) (host and port)")
  }
  region := os.Args[1]
  hostAndPort := os.Args[2]
  ctx := context.Background()
```

```
  res, err := findNode(ctx, region, hostAndPort,
 "72c2e8c1-7d5f-5f30-10ca-9d2bb8c4afbc")
  if err != nil {
    log.Fatal(err)
  }
  fmt.Println(res)
}
```

Comportamento da conexão do Bolt no Neptune

Veja alguns fatos sobre conexões do Bolt no Neptune:

- Como as conexões do Bolt são criadas na camada TCP, você não pode usar um Application Load Balancer na frente delas, como acontece com um endpoint HTTP.
- A porta que o Neptune usa para conexões do Bolt é a porta do cluster de banco de dados.
- Com base no preâmbulo do Bolt passado para ele, o servidor do Neptune seleciona a versão mais atualizada do Bolt (1, 2, 3 ou 4.0).
- O número máximo de conexões com o servidor do Neptune que um cliente pode abrir em qualquer momento é mil.
- Se o cliente não fechar uma conexão após uma consulta, essa conexão poderá ser usada para executar a próxima consulta.
- No entanto, se uma conexão ficar inativa por vinte minutos, o servidor a fechará automaticamente.
- Se a autenticação do IAM não estiver habilitada, você poderá usar `AuthTokens.none()` em vez de fornecer um nome de usuário e senha fictícios. Por exemplo, em Java:

  ```
  GraphDatabase.driver("bolt://(your cluster endpoint URL):(your cluster port)",
   AuthTokens.none(),

   Config.builder().withEncryption().withTrustStrategy(TrustStrategy.trustSystemCertificates())
  ```

- Quando a autenticação do IAM é habilitada, uma conexão do Bolt é sempre desconectada alguns minutos mais de dez dias após ter sido estabelecida, caso ainda não tenha sido fechada por algum outro motivo.
- Se o cliente enviar uma consulta para execução em uma conexão sem ter consumido os resultados de uma consulta anterior, a nova consulta será descartada. Em vez disso, para descartar os resultados anteriores, o cliente deve enviar uma mensagem de redefinição pela conexão.
- Somente uma transação pode ser criada por vez em uma conexão específica.

- Se ocorrer uma exceção durante uma transação, o servidor do Neptune reverterá a transação e fechará a conexão. Nesse caso, o driver cria uma conexão para a próxima consulta.
- Esteja ciente de que as sessões não são seguras para threads. Várias operações paralelas devem usar várias sessões separadas.

Exemplos de consulta parametrizada do openCypher

O Neptune é compatível com consultas parametrizadas do openCypher. Isso permite que você use a mesma estrutura de consulta várias vezes com argumentos diferentes. Como a estrutura da consulta não muda, o Neptune pode armazenar em cache a árvore de sintaxe abstrata (AST) em vez de analisá-la várias vezes.

Exemplo de consulta parametrizada do openCypher usando o endpoint HTTPS

Veja um exemplo de uso de uma consulta parametrizada com o endpoint HTTPS do openCypher no Neptune A consulta é:

```
MATCH (n {name: $name, age: $age})
RETURN n
```

Os parâmetros são definidos da seguinte forma:

```
parameters={"name": "john", "age": 20}
```

Usando `GET`, é possível enviar a consulta parametrizada da seguinte forma:

```
curl -k \
  "https://localhost:8182/openCypher?query=MATCH%20%28n%20%7Bname:\$name,age:\$age%7D
%29%20RETURN%20n&parameters=%7B%22name%22:%22john%22,%22age%22:20%7D"
```

Você também pode usar `POST`:

```
curl -k \
  https://localhost:8182/openCypher \
  -d "query=MATCH (n {name: \$name, age: \$age}) RETURN n" \
  -d "parameters={\"name\": \"john\", \"age\": 20}"
```

Ou usando `DIRECT POST`:

```
curl -k \
   -H "Content-Type: application/opencypher" \
  "https://localhost:8182/openCypher?parameters=%7B%22name%22:%22john%22,%22age
%22:20%7D" \
  -d "MATCH (n {name: \$name, age: \$age}) RETURN n"
```

Exemplos de consulta parametrizada do openCypher usando o Bolt

Veja um exemplo em Python de uma consulta parametrizada do openCypher usando o protocolo Bolt:

```
from neo4j import GraphDatabase
uri = "bolt://[neptune-endpoint-url]:8182"
driver = GraphDatabase.driver(uri, auth=("", ""))

def match_name_and_age(tx, name, age):
  # Parameterized Query
  tx.run("MATCH (n {name: $name, age: $age}) RETURN n", name=name, age=age)

with driver.session() as session:
  # Parameters
  session.read_transaction(match_name_and_age, "john", 20)

driver.close()
```

Veja um exemplo em Java de uma consulta parametrizada do openCypher usando o protocolo Bolt:

```
Driver driver = GraphDatabase.driver("bolt+s://(your cluster endpoint URL):8182");
HashMap<String, Object> parameters = new HashMap<>();
parameters.put("name", "john");
parameters.put("age", 20);
String queryString = "MATCH (n {name: $name, age: $age}) RETURN n";
Result result = driver.session().run(queryString, parameters);
```

Modelo de dados do openCypher

O mecanismo do openCypher no Neptune se baseia no mesmo modelo de grafo de propriedades do Gremlin. Em particular:

- Cada nó tem um ou mais rótulos. Se você inserir um nó sem rótulos, um rótulo padrão chamado `vertex` será anexado. Se você tentar excluir todos os rótulos de um nó, ocorrerá um erro.

- Relacionamento é uma entidade que tem exatamente um tipo de relacionamento e que forma uma conexão unidirecional entre dois nós (ou seja, de um dos nós para o outro).
- Tanto os nós quanto os relacionamentos podem ter propriedades, mas não precisam. O Neptune é compatível com nós e relacionamentos com zero propriedades.
- O Neptune não é compatível com metapropriedades, que também não estão incluídas na especificação do openCypher.
- As propriedades no grafo poderão ter vários valores se tiverem sido criadas usando o Gremlin. Ou seja, um nó ou uma propriedade de relacionamento pode ter um conjunto de valores diferentes em vez de apenas um. O Neptune estendeu a semântica do openCypher para lidar com propriedades de vários valores normalmente.

Os tipos de dados compatíveis estão documentados em Formato de dados do openCypher. No entanto, não recomendamos inserir valores de propriedades `Array` em um grafo do openCypher no momento. Embora seja possível inserir um valor de propriedade de matriz usando o carregador em massa, a versão atual do openCypher no Neptune o trata como um conjunto de propriedades de vários valores em vez de um único valor de lista.

Veja a lista dos tipos de dados compatíveis com esta versão:

- `Bool`
- `Byte`
- `Short`
- `Int`
- `Long`
- `Float` (Inclui infinito positivo e negativo e NaN, mas não INF)
- `Double` (Inclui infinito positivo e negativo e NaN, mas não INF)
- `DateTime`
- `String`

O atributo **`explain`** do openCypher

O atributo `explain` do openCypher é uma ferramenta de autoatendimento no Amazon Neptune que ajuda a compreender a abordagem da execução realizada pelo mecanismo do Neptune. Para

invocar explain, transmita um parâmetro a uma solicitação HTTPS do openCypher com o valor `explain=`*`mode`*, em que o valor *mode* pode ser um dos seguintes:

- **`static`**: no modo `static`, o `explain` imprime somente a estrutura estática do plano de consulta. Na verdade, ele não executa a consulta.
- **`dynamic`**: no modo `dynamic`, `explain` também executa a consulta e inclui aspectos dinâmicos do plano de consulta. Eles podem incluir o número de associações intermediárias que fluem por meio dos operadores, a proporção de associações de entrada para associações de saída e o tempo total utilizado pelos operadores.
- **`details`**: no modo `details`, `explain` imprime as informações mostradas no modo dinâmico mais detalhes adicionais, como a string de consulta do openCypher real e a contagem de intervalo estimada para o padrão subjacente a um operador de junção.

Por exemplo, usando `POST`:

```
curl HTTPS://server:port/openCypher \
  -d "query=MATCH (n) RETURN n LIMIT 1;" \
  -d "explain=dynamic"
```

Ou usando `GET`:

```
curl -X GET \
  "HTTPS://server:port/openCypher?query=MATCH%20(n)%20RETURN%20n%20LIMIT
%201&explain=dynamic"
```

Limitações do **`explain`** do openCypher no Neptune

A versão atual do explain do openCypher tem as seguintes limitações:

- No momento, os planos de explain estão disponíveis apenas para consultas que realizam operações somente leitura. Consultas que realizam qualquer tipo de mutação, como `CREATE`, `DELETE`, `MERGE`, `SET`, etc., não são aceitas.
- Os operadores e as saídas de um plano específico podem mudar em versões futuras.

Operadores do DFE na saída **`explain`** do openCypher

Para usar as informações fornecidas pelo atributo `explain` do openCypher, é necessário entender alguns detalhes sobre como o mecanismo de consulta do DFE funciona (sendo o DFE o mecanismo que o Neptune usa para processar as consultas do openCypher).

O mecanismo do DFE converte cada consulta em um pipeline de operadores. A partir do primeiro operador, soluções intermediárias fluem de um operador até o próximo por meio desse pipeline de operadores. Cada linha da tabela de explain representa um resultado, até o ponto de avaliação.

Os operadores que podem aparecer em um plano de consulta do DFE são os seguintes:

DFEApply: executa a função especificada na seção de argumentos, no valor armazenado na variável especificada

DFE BindRelation — Vincula variáveis com os nomes especificados

DFE ChunkLocal SubQuery — Essa é uma operação sem bloqueio que atua como um invólucro em torno das subconsultas que estão sendo executadas.

DFE DistinctColumn — Retorna o subconjunto distinto dos valores de entrada com base na variável especificada.

DFE DistinctRelation — Retorna o subconjunto distinto das soluções de entrada com base na variável especificada.

DFEDrain: aparece no final de uma subconsulta para atuar como uma etapa de encerramento dessa subconsulta. O número de soluções é registrado como `Units In`. `Units Out` é sempre zero.

DFE ForwardValue — copia todos os blocos de entrada diretamente como blocos de saída para serem passados para seu operador a jusante.

DFE GroupBy HashIndex — Executa uma operação agrupada nas soluções de entrada com base em um índice de hash previamente calculado (usando a operação). `DFEHashIndexBuild` Como saída, a entrada fornecida é estendida por uma coluna contendo uma chave de grupo para cada solução de entrada.

DFE HashIndex Build — Cria um índice de hash sobre um conjunto de variáveis como efeito colateral. Esse índice de hash geralmente é reutilizado em operações posteriores. Consulte `DFEHashIndexJoin` ou `DFEGroupByHashIndex` para saber onde esse índice de hash pode ser usado.

DFE HashIndex Join — Executa uma junção das soluções recebidas em relação a um índice de hash criado anteriormente. Consulte `DFEHashIndexBuild` para saber onde esse índice de hash pode ser criado.

DFE JoinExists — Toma uma relação de entrada à esquerda e à direita e retém valores da relação esquerda que têm um valor correspondente na relação direita, conforme definido pelas variáveis de junção fornecidas.

: é uma operação sem bloqueio que atua como um invólucro de uma subconsulta, permitindo que ela seja executada repetidamente para uso em loops.

DFE MergeChunks — Essa é uma operação de bloqueio que combina partes de seu operador a montante em um único bloco de soluções para passar para o operador a jusante (inverso de). `DFESplitChunks`

DFEMinus: pega uma relação de entrada à esquerda e à direita e retém valores da relação esquerda que não têm um valor correspondente na relação direita, conforme definido pelas variáveis de junção fornecidas. Se não houver sobreposição de variáveis em ambas as relações, este operador simplesmente retornará a relação de entrada à esquerda.

DFE NotExists — Toma uma relação de entrada à esquerda e à direita e retém valores da relação esquerda que não têm um valor correspondente na relação direita, conforme definido pelas variáveis de junção fornecidas. Se não houver sobreposição de variáveis em ambas as relações, este operador retornará uma relação vazia.

DFE OptionalJoin — Executa uma junção externa esquerda (também chamada de junção OPCIONAL): soluções do lado esquerdo que têm pelo menos um parceiro de junção no lado direito são unidas, e soluções do lado esquerdo sem parceiro de junção no lado direito são encaminhadas como estão. Essa é uma operação de bloqueio.

DFE PipelineJoin — une a entrada ao padrão de tupla definido pelo argumento. `pattern`

PipelineRangeContagem de DFE — Conta o número de soluções que correspondem a um determinado padrão e retorna uma única solução única contendo o valor da contagem.

DFE PipelineScan — Verifica o banco de dados em busca do `pattern` argumento fornecido, com ou sem um determinado filtro na (s) coluna (s).

DFEProject: usa várias colunas de entrada e projeta somente as colunas desejadas.

DFEReduce: executa a função de agregação especificada em variáveis determinadas.

DFE RelationalJoin — une a entrada do operador anterior com base nas chaves de padrão especificadas usando uma junção de mesclagem. Essa é uma operação de bloqueio.

DFE RouteChunks — pega pedaços de entrada de sua borda de entrada singular e os direciona ao longo de suas múltiplas bordas de saída.

DFE SelectRows — Esse operador pega seletivamente as linhas de suas soluções de relação de entrada à esquerda para encaminhar para o operador a jusante. As linhas selecionadas com base nos identificadores de linha fornecidos na relação de entrada direita do operador.

DFESerialize: serializa os resultados finais de uma consulta em uma serialização de string JSON, mapeando cada solução de entrada para o nome de variável apropriado. Para resultados de nós e bordas, esses resultados são serializados em um mapa de propriedades e metadados da entidade.

DFESort: pega uma relação de entrada e produz uma relação ordenada com base na chave de classificação fornecida.

SplitByGrupo DFE — divide cada bloco de entrada único de uma borda de entrada em blocos de saída menores correspondentes aos grupos de linhas identificados por IDs de linha do bloco de entrada correspondente da outra borda de entrada.

DFE SplitChunks — Divide cada bloco de entrada em pedaços menores de saída (inverso de). `DFEMergeChunks`

DFE StreamingHash IndexBuild — Versão de streaming do. `DFEHashIndexBuild`

StreamingGroupByHashÍndice DFE — Versão de streaming do. `DFEGroupByHashIndex`

DFESubquery: esse operador aparece no início de todos os planos e encapsula as partes do plano que são executadas no mecanismo do DFE, que é o plano inteiro do openCypher.

DFE SymmetricHash Join — une a entrada do operador anterior com base nas chaves de padrão especificadas usando uma junção de hash. Essa é uma operação sem bloqueio.

DFESync: este operador é um operador de sincronização que oferece suporte a planos sem bloqueio. Ele pega soluções de duas bordas de entrada e as encaminha para as bordas downstream apropriadas. Para fins de sincronização, as entradas ao longo de uma dessas bordas podem ser armazenadas em buffer internamente.

DFETee: é um operador de ramificação que envia o mesmo conjunto de soluções a vários operadores.

DFE TermResolution — Executa uma operação de localização ou globalização em suas entradas, resultando em colunas de identificadores localizados ou globalizados, respectivamente.

: desdobra listas de valores de uma coluna de entrada para a coluna de saída como elementos individuais.

DFEUnion: pega duas ou mais relações de entrada e produz uma união dessas relações usando o esquema de saída desejado.

SolutionInjection— Aparece antes de tudo na saída de explicação, com um valor de 1 na coluna Unidades de saída. No entanto, funciona sem operação e, na verdade, não injeta nenhuma solução no mecanismo do DFE.

TermResolution— Aparece no final dos planos e traduz objetos do motor Neptune em objetos OpenCypher.

Colunas na saída **`explain`** do openCypher

As informações do plano de consulta que o Neptune gera como saída do explain do openCypher contêm tabelas com um operador por linha. A tabela tem as seguintes colunas:

ID: o ID numérico desse operador no plano.

Saída 1 (e Saída 2): os IDs dos operadores posteriores a esse operador. Pode haver no máximo dois operadores posteriores.

Nome: o nome desse operador.

Argumentos: qualquer detalhe relevante para o operador. Isso inclui itens como esquema de entrada, esquema de saída, padrão (para `PipelineScan` e `PipelineJoin`), etc.

Modo: um rótulo que descreve o comportamento fundamental do operador. Essa coluna está quase toda em branco (`-`). Uma exceção é `TermResolution`, em que o modo pode estar `id2value_opencypher`, indicando uma resolução do ID ao valor do openCypher.

Unidades de entrada: o número de soluções transmitidas como entrada para esse operador. Operadores sem operadores anteriores, como `DFEPipelineScan`, `SolutionInjections` e `DFESubquery` sem valor estático injetado, teriam valor zero.

Unidades de saída: o número de soluções produzidas como saída desse operador. `DFEDrain` é um caso especial, em que o número de soluções que estão sendo drenadas é registrado em `Units In` e `Units Out` é sempre zero.

Proporção: a proporção de `Units Out` para `Units In`.

Tempo (ms): o tempo de CPU consumido por esse operador, em milissegundos.

Um exemplo básico de saída de explain do openCypher

Veja um exemplo básico da saída de `explain` do openCypher. A consulta é uma pesquisa de nó único no conjunto de dados de rotas aéreas para um nó com o código do aeroporto `ATL` que invoca `explain` usando o modo `details` no formato de saída ASCII padrão:

```
curl -d "query=MATCH (n {code: 'ATL'}) RETURN n" -k https://localhost:8182/openCypher -
d "explain=details"
                                    ~
Query:
MATCH (n {code: 'ATL'}) RETURN n

################################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments          # Mode                #
 Units In # Units Out # Ratio # Time (ms) #
################################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]     # -                   #
 0        # 1         # 0.00  # 0         #
################################################################################################
# 1  # 2      # -      # DFESubquery       # subQuery=subQuery1 # -                   #
 0        # 1         # 0.00  # 4.00      #
################################################################################################
# 2  # -      # -      # TermResolution    # vars=[?n]          # id2value_opencypher #
 1        # 1         # 1.00  # 2.00      #
################################################################################################


subQuery1
################################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments
                                                                  # Mode # Units
 In # Units Out # Ratio # Time (ms) #
################################################################################################
# 0  # 1      # -      # DFEPipelineScan       # pattern=Node(?n) with property 'code'
 as ?n_code2 and label 'ALL'                                      # -    # 0
   # 1         # 0.00  # 0.21      #
#    #        #        #                      # inlineFilters=[(?n_code2 IN
 ["ATL"^^xsd:string])]                                                  #
  #          #           #       #           #
```

```
#    #        #        #                      # patternEstimate=1
                                                                  #      #
  #           #       #           #
#########################################################################################
# 1  # 2      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#9d84f97c-c3b0-459a-98d5-955a8726b159/graph_1 # -    #
 1        # 1         # 1.00  # 0.04      #
#########################################################################################
# 2  # 3      # -      # DFEProject            # columns=[?n]
                                                                  # -    # 1
  # 1         # 1.00  # 0.04      #
#########################################################################################
# 3  # -      # -      # DFEDrain              # -
                                                                  # -    # 1
  # 0         # 0.00  # 0.03      #
#########################################################################################


subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/graph#9d84f97c-
c3b0-459a-98d5-955a8726b159/graph_1
#########################################################################################
# ID # Out #1 # Out #2 # Name                 # Arguments
                    # Mode # Units In # Units Out # Ratio # Time (ms) #
#########################################################################################
# 0  # 1      # -      # DFESolutionInjection # outSchema=[?n, ?n_code2]
                    # -    # 0        # 1         # 0.00  # 0.02      #
#########################################################################################
# 1  # 2      # 3      # DFETee               # -
                    # -    # 1        # 2         # 2.00  # 0.02      #
#########################################################################################
# 2  # 4      # -      # DFEDistinctColumn    # column=?n
                    # -    # 1        # 1         # 1.00  # 0.20      #
#    #        #        #                      # ordered=false
                    #      #          #           #       #           #
#########################################################################################
# 3  # 5      # -      # DFEHashIndexBuild    # vars=[?n]
                    # -    # 1        # 1         # 1.00  # 0.04      #
#########################################################################################
# 4  # 5      # -      # DFEPipelineJoin      # pattern=Node(?n) with property 'ALL'
 and label '?n_label1' # -    # 1        # 1         # 1.00  # 0.25      #
#    #        #        #                      # patternEstimate=3506
                    #      #          #           #       #           #
#########################################################################################
```

```
# 5  # 6      # 7      # DFESync              # -
                    # -    # 2        # 2         # 1.00  # 0.02      #
#################################################################################################
# 6  # 8      # -      # DFEForwardValue      # -
                    # -    # 1        # 1         # 1.00  # 0.01      #
#################################################################################################
# 7  # 8      # -      # DFEForwardValue      # -
                    # -    # 1        # 1         # 1.00  # 0.01      #
#################################################################################################
# 8  # 9      # -      # DFEHashIndexJoin     # -
                    # -    # 2        # 1         # 0.50  # 0.35      #
#################################################################################################
# 9  # -      # -      # DFEDrain              # -
                    # -    # 1        # 0         # 0.00  # 0.02      #
#################################################################################################
```

No nível superior, `SolutionInjection` aparece antes de tudo, com uma unidade de saída. Observe que, na verdade, ele não injeta nenhuma solução. Você pode ver que o próximo operador, `DFESubquery`, tem 0 unidade de entrada.

Depois de `SolutionInjection`, no nível superior, estão os operadores `DFESubquery` e `TermResolution`. `DFESubquery` encapsula as partes do plano de execução da consulta que está sendo enviado ao mecanismo do DFE (para consultas do openCypher, todo o plano de consulta é executado pelo DFE). Todos os operadores no plano de consulta estão aninhados dentro de `subQuery1` que é referenciado por `DFESubquery`. A única exceção é `TermResolution`, que materializa os IDs internos em objetos do openCypher totalmente serializados.

Todos os operadores que são enviados por push ao mecanismo do DFE têm nomes que começam com o prefixo `DFE`. Conforme mencionado acima, todo o plano de consulta do openCypher é executado pelo DFE, portanto, como resultado, todos os operadores, exceto o operador final `TermResolution`, começam com `DFE`.

Dentro de `subQuery1`, pode haver zero ou mais operadores `DFEChunkLocalSubQuery` ou `DFELoopSubQuery` que encapsulam uma parte do plano de execução enviado que é executado em um mecanismo limitado pela memória. `DFEChunkLocalSubQuery` aqui contém uma `SolutionInjection` que é usada como entrada para a subconsulta. Para encontrar a tabela dessa subconsulta na saída, procure o `subQuery=`*`graph URI`* especificado na coluna `Arguments` para o operador `DFEChunkLocalSubQuery` ou `DFELoopSubQuery`.

Em `subQuery1`, `DFEPipelineScan` com `ID` 0 verifica o banco de dados em busca de um `pattern` especificado. O padrão verifica uma entidade com a propriedade `code` salva como

uma variável `?n_code2` em todos os rótulos (você pode filtrar por um rótulo específico anexando `airport` a `n:airport`). O argumento `inlineFilters` mostra a filtragem da propriedade `code` igualando `ATL`.

Depois, o operador `DFEChunkLocalSubQuery` une os resultados intermediários de uma subconsulta que contém `DFEPipelineJoin`. Isso garante que `?n` seja realmente um nó, já que a `DFEPipelineScan` anterior verifica qualquer entidade com a propriedade `code`.

Exemplo de saída de **`explain`** para uma pesquisa de relacionamento com um limite

Essa consulta procura relacionamentos entre dois nós anônimos com o tipo `route` e gera no máximo dez. Novamente, o modo `explain` é `details` e o formato de saída é ASCII padrão. Confira esta saída `explain`:

Aqui, `DFEPipelineScan` verifica as bordas que começam em um nó anônimo `?anon_node7` e terminam em outro nó anônimo `?anon_node21`, com um tipo de relacionamento salvo como `?p_type1`. Existe um filtro para `?p_type1` ser `el://route` (em que `el` significa o rótulo de borda), que corresponde a `[p:route]` na string de consulta.

`DFEDrain` coleta a solução de saída com um limite de dez, conforme mostrado na coluna `Arguments`. `DFEDrain` termina quando o limite é atingido ou todas as soluções são produzidas, o que ocorrer primeiro.

```
curl -d "query=MATCH ()-[p:route]->() RETURN p LIMIT 10" -k https://localhost:8182/
openCypher -d "explain=details"
                                        ~
Query:
MATCH ()-[p:route]->() RETURN p LIMIT 10

#######################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments          # Mode                #
 Units In # Units Out # Ratio # Time (ms) #
#######################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]     # -                   #
 0        # 1         # 0.00  # 0         #
#######################################################################################
# 1  # 2      # -      # DFESubquery       # subQuery=subQuery1 # -                   #
 0        # 10        # 0.00  # 5.00      #
#######################################################################################
# 2  # -      # -      # TermResolution    # vars=[?p]          # id2value_opencypher #
 10       # 10        # 1.00  # 1.00      #
#######################################################################################
```

```
subQuery1
#################################################################################################
# ID # Out #1 # Out #2 # Name            # Arguments
              # Mode # Units In # Units Out # Ratio # Time (ms) #
#################################################################################################
# 0  # 1      # -      # DFEPipelineScan # pattern=Edge((?anon_node7)-[?p:?p_type1]->(?
anon_node21)) # -    # 0        # 1000      # 0.00  # 0.66      #
#    #        #        #                 # inlineFilters=[(?p_type1 IN [<el://route>])]
              #      #          #           #       #           #
#    #        #        #                 # patternEstimate=26219
              #      #          #           #       #           #
#################################################################################################
# 1  # 2      # -      # DFEProject      # columns=[?p]
              # -    # 1000     # 1000      # 1.00  # 0.14      #
#################################################################################################
# 2  # -      # -      # DFEDrain        # limit=10
              # -    # 1000     # 0         # 0.00  # 0.11      #
#################################################################################################
```

Exemplo de saída de **`explain`** para uma função de expressão de valor

A função é:

```
MATCH (a) RETURN DISTINCT labels(a)
```

Na saída de `explain` abaixo, `DFEPipelineScan` (ID 0) verifica todos os rótulos dos nós. Isso corresponde a `MATCH (a)`.

`DFEChunkLocalSubquery` (ID 1) agrega o rótulo de `?a` de cada `?a`. Isso corresponde a `labels(a)`. É possível ver isso por meio de `DFEApply` e `DFEReduce`.

`BindRelation` (ID 2) é usado para renomear a coluna genérica `?__gen_labelsOfa2` para `?labels(a)`.

`DFEDistinctRelation` (ID 4) recupera somente os rótulos distintos (vários nós :airport forneceriam rótulos duplicados (a): ["airport"]). Isso corresponde a `DISTINCT labels(a)`.

```
curl -d "query=MATCH (a) RETURN DISTINCT labels(a)" -k https://localhost:8182/
openCypher -d "explain=details"
                                          ~
```

```
Query:
MATCH (a) RETURN DISTINCT labels(a)

#################################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments          # Mode                #
 Units In # Units Out # Ratio # Time (ms) #
#################################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]     # -                   #
 0        # 1         # 0.00  # 0         #
#################################################################################################
# 1  # 2      # -      # DFESubquery       # subQuery=subQuery1 # -                   #
 0        # 5         # 0.00  # 81.00     #
#################################################################################################
# 2  # -      # -      # TermResolution    # vars=[?labels(a)]  # id2value_opencypher #
 5        # 5         # 1.00  # 1.00      #
#################################################################################################



subQuery1
#################################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments
                                                                  # Mode # Units
 In # Units Out # Ratio # Time (ms) #
#################################################################################################
# 0  # 1      # -      # DFEPipelineScan       # pattern=Node(?a) with property 'ALL'
 and label '?a_label1'                                              # -    # 0
    # 3750      # 0.00  # 26.77     #
#    #        #        #                       # patternEstimate=3506
                                                                 #      #
  #           #       #           #
#################################################################################################
# 1  # 2      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#8b314f55-2cc7-456a-a48a-c76a0465cfab/graph_1 # -    #
 3750     # 3750      # 1.00  # 0.04      #
#################################################################################################
# 2  # 3      # -      # DFEBindRelation       # inputVars=[?a, ?__gen_labelsOfa2, ?
__gen_labelsOfa2]                                                  # -    # 3750
     # 3750      # 1.00  # 0.08      #
#    #        #        #                       # outputVars=[?a, ?__gen_labelsOfa2, ?
labels(a)]                                                         #      #
    #           #       #           #
#################################################################################################
```

```
# 3  # 4      # -      # DFEProject           # columns=[?labels(a)]
                                                                  # -    # 3750
  # 3750      # 1.00  # 0.05      #
#########################################################################################
# 4  # 5      # -      # DFEDistinctRelation   # -
                                                                  # -    # 3750
  # 5         # 0.00  # 2.78      #
#########################################################################################
# 5  # -      # -      # DFEDrain              # -
                                                                  # -    # 5
  # 0         # 0.00  # 0.03      #
#########################################################################################


subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/graph#8b314f55-2cc7-456a-
a48a-c76a0465cfab/graph_1
#########################################################################################
# ID # Out #1 # Out #2 # Name                # Arguments
                    # Mode     # Units In # Units Out # Ratio # Time (ms) #
#########################################################################################
# 0  # 1      # -      # DFESolutionInjection # outSchema=[?a]
                    # -        # 0        # 3750      # 0.00  # 0.02      #
#########################################################################################
# 1  # 2      # 3      # DFETee               # -
                    # -        # 3750     # 7500      # 2.00  # 0.02      #
#########################################################################################
# 2  # 4      # -      # DFEProject           # columns=[?a]
                    # -        # 3750     # 3750      # 1.00  # 0.04      #
#########################################################################################
# 3  # 17     # -      # DFEOptionalJoin      # -
                    # -        # 7500     # 3750      # 0.50  # 0.44      #
#########################################################################################
# 4  # 5      # -      # DFEDistinctRelation  # -
                    # -        # 3750     # 3750      # 1.00  # 2.23      #
#########################################################################################
# 5  # 6      # -      # DFEDistinctColumn    # column=?a
                    # -        # 3750     # 3750      # 1.00  # 1.50      #
#    #        #        #                     # ordered=false
                    #          #          #           #       #           #
#########################################################################################
# 6  # 7      # -      # DFEPipelineJoin      # pattern=Node(?a) with property 'ALL'
 and label '?a_label3' # -        # 3750     # 3750      # 1.00  # 10.58     #
#    #        #        #                     # patternEstimate=3506
                    #          #          #           #       #           #
```

```
#########################################################################################
# 7  # 8      # 9      # DFETee               # -
                    # -        # 3750     # 7500      # 2.00  # 0.02      #
#########################################################################################
# 8  # 10     # -      # DFEBindRelation      # inputVars=[?a_label3]
                    # -        # 3750     # 3750      # 1.00  # 0.04      #
#    #        #        #                      # outputVars=[?100]
                    #          #          #           #       #           #
#########################################################################################
# 9  # 11     # -      # DFEBindRelation      # inputVars=[?a, ?a_label3, ?100]
                    # -        # 7500     # 3750      # 0.50  # 0.07      #
#    #        #        #                      # outputVars=[?a, ?a_label3, ?100]
                    #          #          #           #       #           #
#########################################################################################
# 10 # 9      # -      # DFETermResolution    # column=?100
                    # id2value # 3750     # 3750      # 1.00  # 7.60      #
#########################################################################################
# 11 # 12     # -      # DFEBindRelation      # inputVars=[?a, ?a_label3, ?100]
                    # -        # 3750     # 3750      # 1.00  # 0.06      #
#    #        #        #                      # outputVars=[?a, ?100, ?a_label3]
                    #          #          #           #       #           #
#########################################################################################
# 12 # 13     # -      # DFEApply             # functor=nodeLabel(?a_label3)
                    # -        # 3750     # 3750      # 1.00  # 0.55      #
#########################################################################################
# 13 # 14     # -      # DFEProject           # columns=[?a, ?a_label3_alias4]
                    # -        # 3750     # 3750      # 1.00  # 0.05      #
#########################################################################################
# 14 # 15     # -      # DFEMergeChunks       # -
                    # -        # 3750     # 3750      # 1.00  # 0.02      #
#########################################################################################
# 15 # 16     # -      # DFEReduce            # functor=collect(?a_label3_alias4)
                    # -        # 3750     # 3750      # 1.00  # 6.37      #
#    #        #        #                      # segmentationKey=[?a]
                    #          #          #           #       #           #
#########################################################################################
# 16 # 3      # -      # DFEMergeChunks       # -
                    # -        # 3750     # 3750      # 1.00  # 0.03      #
#########################################################################################
# 17 # -      # -      # DFEDrain             # -
                    # -        # 3750     # 0         # 0.00  # 0.02      #
#########################################################################################
```

Exemplo de saída de **`explain`** para uma função de expressão matemática

Neste exemplo, `RETURN abs(-10)` executa uma avaliação simples, tomando o valor absoluto de uma constante, `-10`.

`DFEChunkLocalSubQuery` (ID 1) executa uma injeção de solução para o valor estático `-10`, que é armazenado na variável, `?100`.

`DFEApply` (ID 2) é o operador que executa a função de valor absoluto `abs()` no valor estático armazenado na variável `?100`.

Veja a consulta e a saída de `explain` resultante:

```
curl -d "query=RETURN abs(-10)" -k https://localhost:8182/openCypher  -d
 "explain=details"
                                                    ~
Query:
RETURN abs(-10)

################################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments             # Mode
  # Units In # Units Out # Ratio # Time (ms) #
################################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]        # -
  # 0        # 1        # 0.00  # 0         #
################################################################################################
# 1  # 2      # -      # DFESubquery       # subQuery=subQuery1    # -
  # 0        # 1        # 0.00  # 4.00      #
################################################################################################
# 2  # -      # -      # TermResolution    # vars=[?_internalVar1] #
 id2value_opencypher # 1        # 1         # 1.00  # 1.00      #
################################################################################################


subQuery1
################################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments
                                                                    # Mode # Units
 In # Units Out # Ratio # Time (ms) #
################################################################################################
# 0  # 1      # -      # DFESolutionInjection  # outSchema=[]
                                                                  # -    # 0
  # 1         # 0.00  # 0.01      #
```

```
################################################################################################
# 1  # 2      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#c4cc6148-cce3-4561-93c0-deb91f257356/graph_1 # -    #
 1        # 1         # 1.00  # 0.03      #
################################################################################################
# 2  # 3      # -      # DFEApply              # functor=abs(?100)
                                                                          # -    # 1
  # 1         # 1.00  # 0.26      #
################################################################################################
# 3  # 4      # -      # DFEBindRelation       # inputVars=[?_internalVar2, ?
_internalVar2]                                                             # -
 # 1        # 1         # 1.00  # 0.04      #
#    #        #        #                       # outputVars=[?_internalVar2, ?
_internalVar1]                                                           #
 #          #           #       #           #
################################################################################################
# 4  # 5      # -      # DFEProject            # columns=[?_internalVar1]
                                                                          # -    # 1
  # 1         # 1.00  # 0.06      #
################################################################################################
# 5  # -      # -      # DFEDrain              # -
                                                                          # -    # 1
  # 0         # 0.00  # 0.05      #
################################################################################################

subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/graph#c4cc6148-
cce3-4561-93c0-deb91f257356/graph_1
################################################################################################
# ID # Out #1 # Out #2 # Name                 # Arguments                          #
 Mode # Units In # Units Out # Ratio # Time (ms) #
################################################################################################
# 0  # 1      # -      # DFESolutionInjection # solutions=[?100 -> [-10^^<LONG>]]   # -
    # 0        # 1         # 0.00  # 0.01      #
#    #        #        #                       # outSchema=[?100]                   #
    #          #           #       #           #
################################################################################################
# 1  # 3      # -      # DFERelationalJoin    # joinVars=[]                        # -
    # 2        # 1         # 0.50  # 0.18      #
################################################################################################
# 2  # 1      # -      # DFESolutionInjection # outSchema=[]                       # -
    # 0        # 1         # 0.00  # 0.01      #
################################################################################################
# 3  # -      # -      # DFEDrain              # -                                  # -
    # 1        # 0         # 0.00  # 0.02      #
```

```
################################################################################################
```

Exemplo de saída de **`explain`** para uma consulta de caminho de comprimento variável (VLP)

Este é um exemplo de um plano de consulta mais complexo que lida com uma consulta de caminho de comprimento variável. Este exemplo mostra apenas parte da saída de `explain`, para fins de clareza.

Em `subQuery1`, `DFEPipelineScan` (ID 0) e `DFEChunkLocalSubQuery` (ID 1), que injetam a subconsulta `...graph_1`, são responsáveis pela verificação de um nó com o código `YPO`.

Em `subQuery1`, `DFEChunkLocalSubQuery` (ID 2), que injeta a subconsulta `...graph_2`, é responsável pela verificação de um nó com o código `LAX`.

Em `subQuery1`, `DFEChunkLocalSubQuery` (ID 3) injeta a subconsulta `...graph3`, que contém `DFELoopSubQuery` (ID 17) que, por sua vez, injeta a subconsulta `...graph5`. Essa operação é responsável por resolver o padrão de comprimento variável `-[*2]->` na string de consulta entre dois nós.

```
curl -d "query=MATCH p=(a {code: 'YPO'})-[*2]->(b{code: 'LAX'}) return p" -k https://
localhost:8182/openCypher -d "explain=details"

                                                ~
Query:
MATCH p=(a {code: 'YPO'})-[*2]->(b{code: 'LAX'}) return p

################################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments          # Mode                #
 Units In # Units Out # Ratio # Time (ms) #
################################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]     # -                   #
 0        # 1         # 0.00  # 0         #
################################################################################################
# 1  # 2      # -      # DFESubquery       # subQuery=subQuery1 # -                   #
 0        # 0         # 0.00  # 84.00     #
################################################################################################
# 2  # -      # -      # TermResolution    # vars=[?p]          # id2value_opencypher #
 0        # 0         # 0.00  # 0         #
################################################################################################
```

```
subQuery1
#######################################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments
                                                                                  # Mode # Units
 In # Units Out # Ratio # Time (ms) #
#######################################################################################################
# 0  # 1      # -      # DFEPipelineScan       # pattern=Node(?a) with property 'code'
 as ?a_code7 and label 'ALL'                                                      # -    # 0
   # 1         # 0.00  # 0.68      #
#    #        #        #                       # inlineFilters=[(?a_code7 IN
 ["YPO"^^xsd:string])]                                                                 #
  #          #           #       #           #
#    #        #        #                       # patternEstimate=1
                                                                                #      #
  #           #       #           #
#######################################################################################################
# 1  # 2      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-bbe3-9e99558eca46/graph_1 # -    #
 1        # 1         # 1.00  # 0.03      #
#######################################################################################################
# 2  # 3      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-bbe3-9e99558eca46/graph_2 # -    #
 1        # 1         # 1.00  # 0.02      #
#######################################################################################################
# 3  # 4      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-bbe3-9e99558eca46/graph_3 # -    #
 1        # 0         # 0.00  # 0.04      #
#######################################################################################################
# 4  # 5      # -      # DFEBindRelation       # inputVars=[?__gen_path6, ?
anon_rel26, ?b_code8, ?b, ?a_code7, ?a, ?__gen_path6]                                  # -
   # 0        # 0         # 0.00  # 0.10      #
#    #        #        #                       # outputVars=[?__gen_path6, ?
anon_rel26, ?b_code8, ?b, ?a_code7, ?a, ?p]                                           #
  #          #           #       #           #
#######################################################################################################
# 5  # 6      # -      # DFEProject            # columns=[?p]
                                                                                # -    # 0
  # 0         # 0.00  # 0.05      #
#######################################################################################################
# 6  # -      # -      # DFEDrain              # -
                                                                                # -    # 0
  # 0         # 0.00  # 0.02      #
#######################################################################################################
```

```
subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-
bbe3-9e99558eca46/graph_1
#########################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments
                  # Mode # Units In # Units Out # Ratio # Time (ms) #
#########################################################################################
# 0  # 1      # -      # DFESolutionInjection # outSchema=[?a, ?a_code7]
                  # -    # 0        # 1         # 0.00  # 0.01      #
#########################################################################################
# 1  # 2      # 3      # DFETee               # -
                  # -    # 1        # 2         # 2.00  # 0.01      #
#########################################################################################
# 2  # 4      # -      # DFEDistinctColumn    # column=?a
                  # -    # 1        # 1         # 1.00  # 0.25      #
#    #        #        #                      # ordered=false
                  #      #          #         #       #           #
#########################################################################################
# 3  # 5      # -      # DFEHashIndexBuild    # vars=[?a]
                  # -    # 1        # 1         # 1.00  # 0.05      #
#########################################################################################
# 4  # 5      # -      # DFEPipelineJoin      # pattern=Node(?a) with property 'ALL'
 and label '?a_label1' # -    # 1        # 1         # 1.00  # 0.47      #
#    #        #        #                      # patternEstimate=3506
                  #      #          #         #       #           #
#########################################################################################
# 5  # 6      # 7      # DFESync              # -
                  # -    # 2        # 2         # 1.00  # 0.04      #
#########################################################################################
# 6  # 8      # -      # DFEForwardValue      # -
                  # -    # 1        # 1         # 1.00  # 0.01      #
#########################################################################################
# 7  # 8      # -      # DFEForwardValue      # -
                  # -    # 1        # 1         # 1.00  # 0.01      #
#########################################################################################
# 8  # 9      # -      # DFEHashIndexJoin     # -
                  # -    # 2        # 1         # 0.50  # 0.26      #
#########################################################################################
# 9  # -      # -      # DFEDrain             # -
                  # -    # 1        # 0         # 0.00  # 0.02      #
#########################################################################################
```

```
subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-
bbe3-9e99558eca46/graph_2
#########################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments
                           # Mode # Units In # Units Out # Ratio # Time (ms) #
#########################################################################################
# 0  # 1      # -      # DFEPipelineScan      # pattern=Node(?b) with property 'code'
 as ?b_code8 and label 'ALL' # -    # 0        # 1         # 0.00  # 0.38      #
#    #        #        #                      # inlineFilters=[(?b_code8 IN
 ["LAX"^^xsd:string])]                #      #          #           #       #
  #
#    #        #        #                      # patternEstimate=1
                           #      #          #           #       #           #
#########################################################################################
# 1  # 2      # -      # DFEMergeChunks       # -
                           # -    # 1        # 1         # 1.00  # 0.02      #
#########################################################################################
# 2  # 4      # -      # DFERelationalJoin    # joinVars=[]
                           # -    # 2        # 1         # 0.50  # 0.19      #
#########################################################################################
# 3  # 2      # -      # DFESolutionInjection # outSchema=[?a, ?a_code7]
                           # -    # 0        # 1         # 0.00  # 0         #
#########################################################################################
# 4  # -      # -      # DFEDrain             # -
                           # -    # 1        # 0         # 0.00  # 0.01      #
#########################################################################################


subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-
bbe3-9e99558eca46/graph_3
#########################################################################################
# ID # Out #1 # Out #2 # Name                   # Arguments
                                                                        # Mode     #
 Units In # Units Out # Ratio # Time (ms) #
#########################################################################################
...
# 17 # 18     # -      # DFELoopSubQuery       # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-bbe3-9e99558eca46/graph_5 # -
  # 1        # 2         # 2.00  # 0.31      #
...
```

Transações em openCypher no Neptune

A implementação do openCypher no Amazon Neptune usa a semântica de transação definida pelo Neptune. No entanto, os níveis de isolamento fornecidos pelo driver Bolt têm algumas implicações específicas para a semântica de transações Bolt, conforme descrito nas seções abaixo.

Consultas de transações do Bolt somente leitura

Há várias maneiras pelas quais as consultas somente leitura podem ser processadas, com diferentes modelos de transação e níveis de isolamento, da seguinte forma:

Consultas de transação implícitas somente leitura

Veja um exemplo de transação implícita somente leitura:

```
public void executeReadImplicitTransaction()
{
  // end point
  final String END_POINT = "(End Point URL)";

  // read query
  final String READ_QUERY = "MATCH (n) RETURN n limit 10";

  // create the driver
  final Driver driver = GraphDatabase.driver(END_POINT, AuthTokens.none(),
          Config.builder().withEncryption()
                          .withTrustStrategy(TrustStrategy.trustSystemCertificates())
                          .build());

  // create the session config
  SessionConfig sessionConfig = SessionConfig.builder()
                                             .withFetchSize(1000)
                                             .withDefaultAccessMode(AccessMode.READ)
                                             .build();

  // run the query as access mode read
  driver.session(sessionConfig).readTransaction(new TransactionWork<String>()
    {
      final StringBuilder resultCollector = new StringBuilder();

      @Override
      public String execute(final Transaction tx)
```

```
      {
        // execute the query
        Result queryResult = tx.run(READ_QUERY);

        // Read the result
        for (Record record : queryResult.list())
        {
          for (String key : record.keys())
          {
            resultCollector.append(key)
                           .append(":")
                           .append(record.get(key).asNode().toString());
          }
        }
        return resultCollector.toString();
      }

    }
  );

  // close the driver.
  driver.close();
}
```

Como as réplicas de leitura aceitam somente consultas somente leitura, todas as consultas em réplicas de leitura são executadas como transações implícitas de leitura, independentemente do modo de acesso definido na configuração da sessão. O Neptune avalia as transações implícitas de leitura como consultas somente leitura sob a semântica de isolamento SNAPSHOT.

Em caso de falha, as transações implícitas de leitura são repetidas por padrão.

Consultas de transação somente leitura de confirmação automática

Veja um exemplo de transação de confirmação automática de somente leitura:

```
public void executeAutoCommitTransaction()
{
  // end point
  final String END_POINT = "(End Point URL)";

  // read query
  final String READ_QUERY = "MATCH (n) RETURN n limit 10";
```

```
  // Create the session config.
  final SessionConfig sessionConfig = SessionConfig
    .builder()
    .withFetchSize(1000)
    .withDefaultAccessMode(AccessMode.READ)
    .build();

  // create the driver
  final Driver driver = GraphDatabase.driver(END_POINT, AuthTokens.none(),
    Config.builder()
          .withEncryption()
          .withTrustStrategy(TrustStrategy.trustSystemCertificates())
          .build());

  // result collector
  final StringBuilder resultCollector = new StringBuilder();

  // create a session
  final Session session = driver.session(sessionConfig);

  // run the query
  final Result queryResult = session.run(READ_QUERY);
  for (final Record record : queryResult.list())
  {
    for (String key : record.keys())
    {
      resultCollector.append(key)
                     .append(":")
                     .append(record.get(key).asNode().toString());
    }
  }

  // close the session
  session.close();

  // close the driver
  driver.close();
}
```

Se o modo de acesso estiver definido como `READ` na configuração da sessão, o Neptune avaliará as consultas de transação de confirmação automática como consultas somente leitura sob a semântica de isolamento `SNAPSHOT`. Observe que as réplicas de leitura só aceitam consultas somente leitura.

Se você não transmitir uma configuração de sessão, as consultas de confirmação automática serão processadas por padrão com isolamento de consulta de mutação, portanto, é importante transmitir uma configuração de sessão que defina explicitamente o modo de acesso como `READ`.

Em caso de falha, as consultas de confirmação automática somente leitura não são repetidas.

Consultas de transação explícitas somente leitura

Veja um exemplo de transação explícita somente leitura:

```
public void executeReadExplicitTransaction()
{
  // end point
  final String END_POINT = "(End Point URL)";

  // read query
  final String READ_QUERY = "MATCH (n) RETURN n limit 10";

  // Create the session config.
  final SessionConfig sessionConfig = SessionConfig
    .builder()
    .withFetchSize(1000)
    .withDefaultAccessMode(AccessMode.READ)
    .build();

  // create the driver
  final Driver driver = GraphDatabase.driver(END_POINT, AuthTokens.none(),
    Config.builder()
          .withEncryption()
          .withTrustStrategy(TrustStrategy.trustSystemCertificates())
          .build());

  // result collector
  final StringBuilder resultCollector = new StringBuilder();

  // create a session
  final Session session = driver.session(sessionConfig);

  // begin transaction
  final Transaction tx = session.beginTransaction();

  // run the query on transaction
  final List<Record> list = tx.run(READ_QUERY).list();
```

```
  // read the result
  for (final Record record : list)
  {
    for (String key : record.keys())
    {
      resultCollector
        .append(key)
        .append(":")
        .append(record.get(key).asNode().toString());
    }
  }

  // commit the transaction and for rollback we can use beginTransaction.rollback();
  tx.commit();

  // close the driver
  driver.close();
}
```

Se o modo de acesso estiver definido como `READ` na configuração da sessão, o Neptune avaliará as consultas de transação explícitas somente leitura como consultas somente leitura sob a semântica de isolamento `SNAPSHOT`. Observe que as réplicas de leitura só aceitam consultas somente leitura.

Se você não transmitir uma configuração de sessão, as transações explícitas somente leitura serão processadas por padrão com isolamento de consulta de mutação, portanto, é importante transmitir uma configuração de sessão que defina explicitamente o modo de acesso como `READ`.

Em caso de falha, as consultas explícitas somente leitura são repetidas por padrão.

Consultas de transação do Bolt de mutação

Assim como nas consultas somente leitura, há várias maneiras pelas quais as consultas de mutação podem ser processadas, com diferentes modelos de transação e níveis de isolamento, da seguinte forma:

Consultas de transação de mutação implícitas

Veja um exemplo de transação de mutação implícita:

```
public void executeWriteImplicitTransaction()
{
  // end point
  final String END_POINT = "(End Point URL)";
```

```
// create node with label as label and properties.
final String WRITE_QUERY = "CREATE (n:label {name : 'foo'})";

// Read the vertex created with label as label.
final String READ_QUERY = "MATCH (n:label) RETURN n";

// create the driver
final Driver driver = GraphDatabase.driver(END_POINT, AuthTokens.none(),
  Config.builder()
        .withEncryption()
        .withTrustStrategy(TrustStrategy.trustSystemCertificates())
        .build());

// create the session config
SessionConfig sessionConfig = SessionConfig
  .builder()
  .withFetchSize(1000)
  .withDefaultAccessMode(AccessMode.WRITE)
  .build();

final StringBuilder resultCollector = new StringBuilder();

// run the query as access mode write
driver.session(sessionConfig).writeTransaction(new TransactionWork<String>()
{
  @Override
  public String execute(final Transaction tx)
  {
    // execute the write query and consume the result.
    tx.run(WRITE_QUERY).consume();

    // read the vertex written in the same transaction
    final List<Record> list = tx.run(READ_QUERY).list();

    // read the result
    for (final Record record : list)
    {
      for (String key : record.keys())
      {
        resultCollector
          .append(key)
          .append(":")
          .append(record.get(key).asNode().toString());
```

```
        }
      }
      return resultCollector.toString();
    }
  }); // at the end, the transaction is automatically committed.

  // close the driver.
  driver.close();
}
```

As leituras feitas como parte das consultas de mutação são executadas sob isolamento `READ COMMITTED` com as garantias usuais para transações de mutação do Neptune.

Independentemente de você transmitir especificamente uma configuração de sessão, a transação é sempre tratada como uma transação de gravação.

Sobre conflitos, consulte Resolução de conflitos usando tempos limite de espera de bloqueio.

Consultas de transação de mutação de confirmação automática

As consultas de confirmação automática de mutação herdam o mesmo comportamento das transações implícitas de mutação.

Se você não transmitir uma configuração de sessão, a transação será tratada como uma transação de gravação por padrão.

Em caso de falha, as consultas de confirmação automática de mutação não são repetidas automaticamente.

Consultas de transação de mutação explícitas

Veja um exemplo de transação de mutação explícita:

```
public void executeWriteExplicitTransaction()
{
  // end point
  final String END_POINT = "(End Point URL)";

  // create node with label as label and properties.
  final String WRITE_QUERY = "CREATE (n:label {name : 'foo'})";

  // Read the vertex created with label as label.
  final String READ_QUERY = "MATCH (n:label) RETURN n";
```

```
// create the driver
final Driver driver = GraphDatabase.driver(END_POINT, AuthTokens.none(),
  Config.builder()
        .withEncryption()
        .withTrustStrategy(TrustStrategy.trustSystemCertificates())
        .build());

// create the session config
SessionConfig sessionConfig = SessionConfig
  .builder()
  .withFetchSize(1000)
  .withDefaultAccessMode(AccessMode.WRITE)
  .build();

final StringBuilder resultCollector = new StringBuilder();

final Session session = driver.session(sessionConfig);

// run the query as access mode write
final Transaction tx = driver.session(sessionConfig).beginTransaction();

// execute the write query and consume the result.
tx.run(WRITE_QUERY).consume();

// read the result from the previous write query in a same transaction.
final List<Record> list = tx.run(READ_QUERY).list();

// read the result
for (final Record record : list)
{
  for (String key : record.keys())
  {
    resultCollector
      .append(key)
      .append(":")
      .append(record.get(key).asNode().toString());
  }
}

// commit the transaction and for rollback we can use tx.rollback();
tx.commit();

// close the session
session.close();
```

```
  // close the driver.
  driver.close();
}
```

As consultas de mutação explícitas herdam o mesmo comportamento das transações de mutação implícitas.

Se você não transmitir uma configuração de sessão, a transação será tratada como uma transação de gravação por padrão.

Sobre conflitos, consulte Resolução de conflitos usando tempos limite de espera de bloqueio.

Restrições do openCypher no Neptune

A versão do openCypher no Amazon Neptune ainda não é compatível com tudo o que está especificado na Cypher Query Language Reference, Version 9, conforme detalhado em Conformidade com a especificação OpenCypher Espera-se que versões futuras resolvam muitas dessas limitações.

Exceções do openCypher no Neptune

Ao trabalhar com o openCypher no Amazon Neptune, várias exceções podem ocorrer. Veja as exceções comuns que você pode receber, seja do endpoint HTTPS ou do driver Bolt (todas as exceções do driver Bolt são relatadas como exceções do estado do servidor):

Código HTTP	Mensagem de erro	Repetíveis?	Solução	
400	(erro de sintaxe, propagado diretamente do analisador do openCypher)	Não	Corrija a sintaxe da consulta e tente novamente .	
500	`Operation terminated (out of memory)`	Sim	Revise a consulta para adicionar outros critérios de	

Código HTTP	Mensagem de erro	Repetíveis?	Solução	
			filtragem a fim de reduzir a memória necessária.	
500	Operação encerrada (prazo excedido)	Sim	Aumente o tempo limite da consulta no grupo de parâmetros do cluster de banco de dados ou repita a solicitação.	
500	Operação encerrada (cancelada pelo usuário)	Sim	Repetir a solicitação .	
500	A redefinição do banco de dados está em andamento . Tente fazer a consulta novamente depois que o cluster estiver disponível.	Sim	Tente novamente quando a redefinição for concluída.	

Código HTTP	Mensagem de erro	Repetíveis?	Solução
500	A operação falhou devido a operações simultâneas conflitantes (tente novamente). No momento, as transações estão sendo revertidas.	Sim	Tente novamente usando uma estratégia de recuo exponencial e nova tentativa.
400	Exceção de operação/atributo *(nome da operação)* não compatível	Não	A operação especificada não é comportada.
400	Tentativa de atualização do openCypher em uma réplica somente leitura	Não	Altere o ponto final de destino para o ponto final de gravador.
400	MalformedQueryException (Neptune não mostra o estado interno do analisador)	Não	Corrija a sintaxe da consulta e tente novamente.

Código HTTP	Mensagem de erro	Repetíveis?	Solução	
400	Não é possível excluir o nó, porque ele ainda tem relaciona mentos. Para excluir esse nó, é necessário primeiro excluir seus relaciona mentos.	Não	Em vez de utilizar `MATCH (n) DELETE n`, use `MATCH(n) DETACH DELETE(n)` .	

Código HTTP	Mensagem de erro	Repetíveis?	Solução	
400	Operação inválida: tentativa de remover o último rótulo de um nó. Um nó deve ter pelo menos um rótulo.	Não	O Neptune exige que todos os nós tenham pelo menos um rótulo e, se os nós forem criados sem um rótulo explícito, um rótulo padrão `vertex` será atribuído. Altere a lógica da consulta e/ou da aplicação para não excluir o último rótulo. Um rótulo singleton de um nó pode ser atualizado definindo um novo rótulo e, em seguida, removendo o rótulo antigo.	

Código HTTP	Mensagem de erro	Repetíveis?	Solução	
500	Número máximo de solicitaç ões violadas, Configure dQueueCapacity = {} para ConniD = {}	Sim	No momento, apenas 8.192 solicitações simultâneas podem ser processadas, independe ntemente da pilha e do protocolo.	
500	Limite máximo de conexão violado.	Sim	Somente mil conexões simultâneas do Bolt por instância são permitidas (para HTTP, não há limite).	
400	Esperava [um de: Nó, Relacionamento ou Caminho] e recebi um Literal	Não	Confira se você está transmitindo os argumento s corretos, a sintaxe de consulta correta e tente novamente.	

Código HTTP	Mensagem de erro	Repetíveis?	Solução	
400	O valor da propriedade deve ser um literal simples. Ou: mapa esperado para propriedades de Set, mas não encontrei nenhum.	Não	Uma cláusula SET aceita somente literais simples, não tipos compostos.	
400	A entidade encontrada transmitida para exclusão não foi encontrada.	Não	Confira se a entidade que você está tentando excluir existe no banco de dados.	
400	O usuário não tem acesso ao banco de dados.	Não	Confira a política sobre o perfil do IAM que está sendo usada.	
400	Não há token transmitido como parte da solicitação.	Não	Um token assinado corretamente deve ser transmitido como parte da solicitação de consulta em um cluster habilitado para IAM.	

Código HTTP	Mensagem de erro	Repetíveis?	Solução	
400	A mensagem de erro é propagada .	Não	Entre em contato com o AWS Support com o ID da solicitação.	
500	Operação encerrada (erro interno)	Sim	Entre em contato com o AWS Support com o ID da solicitação.	

Acessar o grafo do Neptune com o SPARQL

O SPARQL é uma linguagem de consulta para o Resource Description Framework (RDF), que é um formato de dados de grafos projetado para a web. O Amazon Neptune é compatível com o SPARQL 1.1. Isso significa que você pode se conectar a uma instância de banco de dados do Neptune e consultar o grafo usando a linguagem de consulta descrita na especificação SPARQL 1.1 Query Language.

Uma consulta no SPARQL consiste em uma cláusula `SELECT` para especificar as variáveis a serem retornadas e uma cláusula `WHERE` para especificar quais dados corresponder no gráfico. Se não estiver familiarizado com as consultas do SPARQL, consulte Writing Simple Queries no SPARQL 1.1 Query Language.

> **Important**
>
> Para carregar dados, `SPARQL UPDATE INSERT` pode funcionar bem para um pequeno conjunto de dados, mas se for necessário carregar uma quantidade significativa de dados de um arquivo, consulte Usar o carregador em massa do Amazon Neptune para ingerir dados.

Para obter mais informações sobre as especificidades da implementação do SPARQL do Neptune, consulte Conformidade com padrões do SPARQL.

Antes de começar, você deve ter o seguinte:

- Uma instância de banco de dados do Neptune. Para obter informações sobre como criar uma instância de banco de dados do Neptune, consulte Criar um cluster de banco de dados do Neptune.
- A instância do Amazon EC2 na mesma nuvem privada virtual (VPC) que a instância de banco de dados do Neptune.

Tópicos

Usar o console do RDF4J para conectar-se a uma instância de banco de dados do Neptune

O console RDF4J permite que você experimente gráficos e consultas do Resource Description Framework (RDF) em um ambiente REPL (loop). read-eval-print

Você pode adicionar um banco de dados de gráficos remoto como um repositório e consultá-lo no console do RDF4J. Esta seção descreve a configuração do Console do RDF4J para conectar-se remotamente a uma instância de banco de dados do Neptune.

Como conectar-se ao Neptune usando o console do RDF4J

1. Faça download do SDK do RDF4J na página de download no site do RDF4J.

2. Descompacte o arquivo .zip do SDK do RDF4J.
3. Em um terminal, navegue até o diretório do SDK do RDF4J e digite o comando a seguir para executar o console do RDF4J:

```
bin/console.sh
```

Você deve ver saída semelhante a:

```
14:11:51.126 [main] DEBUG o.e.r.c.platform.PlatformFactory - os.name = linux
14:11:51.130 [main] DEBUG o.e.r.c.platform.PlatformFactory - Detected Posix
 platform
Connected to default data directory
RDF4J Console 3.6.1

3.6.1
Type 'help' for help.
>
```

Agora você está no prompt do >. Esse é o prompt geral do console do RDF4J. Você usa esse prompt para configurar repositórios e outras operações. Um repositório tem seu próprio prompt para execução de consultas.

4. No prompt >, digite o seguinte para criar um repositório do SPARQL para a instância de banco de dados do Neptune:

```
create sparql
```

5. O console do RDF4J solicita valores para as variáveis necessárias para a conexão ao endpoint do SPARQL.

```
Please specify values for the following variables:
```

Especifique os seguintes valores:

Nome da variável	Valor
Endpoint de consulta do SPARQL	`https://`*`your-neptune-endpoint`* `:`*`port`*`/sparql`

Endpoint de atualização do SPARQL	`https://your-neptune-endpoint :port/sparql`
ID do repositório local [endpoint@localhost]	`neptune`
Titulo do repositório [@localhost do repositório do endpoint do SPARQL]	`Neptune DB instance`

Para obter informações sobre como localizar o endereço da instância de banco de dados do Neptune, consulte a seção Conectar-se a endpoints do Amazon Neptune.

Se a operação for bem-sucedida, você verá a seguinte mensagem:

```
Repository created
```

6. No prompt >, insira o seguinte para conectar-se à instância de banco de dados do Neptune:

```
open neptune
```

Se a operação for bem-sucedida, você verá a seguinte mensagem:

```
Opened repository 'neptune'
```

Agora você está no prompt do `neptune>`. Nesse prompt, é possível executar consultas no grafo do Neptune.

> **Note**
>
> Agora que você adicionou o repositório, na próxima vez que executar o `bin/console.sh`, poderá executar o comando `open neptune` imediatamente para conectar-se à instância de banco de dados do Neptune.

7. No prompt `neptune>`, digite o seguinte para executar uma consulta do SPARQL que retorna até 10 dos trios (subject-predicate-object) no gráfico usando a consulta `?s ?p ?o` com um limite de

10. Para consultar outros elementos, substitua o texto depois do comando `sparql` com outra consulta do SPARQL.

```
sparql select ?s ?p ?o where {?s ?p ?o} limit 10
```

Usar o RDF4J Workbench para conectar-se a uma instância de banco de dados do Neptune

Esta seção descreve como conectar-se a uma instância de banco de dados do Amazon Neptune usando o RDF4J Workbench e o RDF4J Server. O RDF4J Server é necessário porque ele atua como um proxy entre o endpoint REST HTTP do SPARQL do Neptune e o RDF4J Workbench.

O RDF4J Workbench fornece uma interface fácil para experimentar com um gráfico, incluindo o carregamento dos arquivos locais. Para obter informações, consulte a Add section na documentação do RDF4J.

Pré-requisitos

Antes de começar, faça o seguinte:

- Instale o Java 1.8 ou posterior.
- Instale o RDF4J Server e o RDF4J Workbench. Para obter informações, consulte Installing RDF4J Server and RDF4J Workbench.

Como usar o RDF4J Workbench para conectar-se ao Neptune

1. Em um navegador da Web, navegue até a URL em que o aplicativo web do RDF4J Workbench está implantado. Por exemplo, se você estiver usando o Apache Tomcat, a URL será: https://*ec2_hostname*:8080/rdf4j-workbench/.
2. Se você for solicitado a Connect to RDF4J Server (Conectar ao RDF4J Server), verifique se o RDF4J Server está instalado, em execução, e que o URL do servidor está correto. Em seguida, prossiga para a próxima etapa.
3. No painel à esquerda, escolha New repository (Novo repositório).

   Em New repository (Novo repositório):

- Na lista suspensa Type (Tipo), escolha SPARQL endpoint proxy (Proxy do endpoint do SPARQL).
- Em ID, digite neptune.
- Em Título, digite Instância de banco de dados do Neptune.

Escolha Próximo.

4. Em New repository (Novo repositório):

   - Em SPARQL query endpoint URL (URL do endpoint de consulta do SPARQL), digite `https://your-neptune-endpoint:port/sparql`.
   - Em SPARQL update endpoint URL (URL do endpoint de atualização do SPARQL), digite `https://your-neptune-endpoint:port/sparql`.

   Para obter informações sobre como localizar o endereço da instância de banco de dados do Neptune, consulte a seção Conectar-se a endpoints do Amazon Neptune.

   Escolha Criar.

5. O repositório do neptune agora é exibido na lista de repositórios. Pode levar alguns minutos para que você possa usar o novo repositório.
6. Na coluna Id da tabela, escolha o link neptune.
7. No painel à esquerda, escolha Query (Consultar).

> **Note**
>
> Se os itens de menu em Explore (Explorar) estiverem desabilitados, poderá ser necessário reconectar-se ao RDF4J Server e escolher o repositório do neptune novamente.
> Você pode fazer isso usando os links de [change] (alterar) no canto superior direito.

8. No campo de consulta, digite a seguinte consulta do SPARQL e, em seguida, selecione Execute (Executar).

```
select ?s ?p ?o where {?s ?p ?o} limit 10
```

O exemplo anterior retorna até 10 dos triplos (subject-predicate-object) no gráfico usando a consulta `?s ?p ?o` com um limite de 10.

Usar o Java para conectar-se a uma instância de banco de dados do Neptune

Esta seção descreve a execução de um exemplo completo de Java que se conecta a uma instância de banco de dados do Amazon Neptune e executa uma consulta do SPARQL.

Siga estas instruções em uma instância do Amazon EC2 na mesma nuvem privada virtual (VPC) que a instância de banco de dados do Neptune.

Como conectar-se ao Neptune usando o Java

1. Instale o Apache Maven na instância do EC2. Primeiro, insira o seguinte para adicionar um repositório com um pacote do Maven:

   ```
   sudo wget https://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-apache-maven.repo
   ```

   Insira o seguinte para definir o número da versão para os pacotes:

   ```
   sudo sed -i s/\$releasever/6/g /etc/yum.repos.d/epel-apache-maven.repo
   ```

   Em seguida, você pode usar yum para instalar o Maven:

   ```
   sudo yum install -y apache-maven
   ```

2. Este exemplo foi testado apenas com o Java 8. Insira o seguinte para instalar o Java 8 na instância do EC2:

   ```
   sudo yum install java-1.8.0-devel
   ```

3. Insira o seguinte para definir o Java 8 como o tempo de execução padrão na instância do EC2:

   ```
   sudo /usr/sbin/alternatives --config java
   ```

   Quando solicitado, insira o número do Java 8.

4. Insira o seguinte para definir o Java 8 como o compilador padrão na instância do EC2:

```
sudo /usr/sbin/alternatives --config javac
```

Quando solicitado, insira o número do Java 8.

5. Em um novo diretório, crie um arquivo `pom.xml` e abra-o em um editor de texto.
6. Copie o seguinte no arquivo `pom.xml` e salve-o (geralmente é possível ajustar os números da versão para a versão estável mais recente):

```
<project xmlns="https://maven.apache.org/POM/4.0.0" xmlns:xsi="https://
www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="https://maven.apache.org/POM/4.0.0 https://maven.apache.org/
maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.amazonaws</groupId>
  <artifactId>RDFExample</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>RDFExample</name>
  <url>https://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>org.eclipse.rdf4j</groupId>
      <artifactId>rdf4j-runtime</artifactId>
      <version>3.6</version>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
          <groupId>org.codehaus.mojo</groupId>
          <artifactId>exec-maven-plugin</artifactId>
          <version>1.2.1</version>
          <configuration>
            <mainClass>com.amazonaws.App</mainClass>
          </configuration>
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
```

```
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

> **Note**
>
> Se estiver modificando um projeto existente do Maven, a dependência necessária será destacada no código anterior.

7. Para criar subdiretórios para o código-fonte de exemplo (`src/main/java/com/amazonaws/`), digite o seguinte na linha de comando:

```
mkdir -p src/main/java/com/amazonaws/
```

8. No diretório `src/main/java/com/amazonaws/`, crie um arquivo denominado `App.java` e abra-o em um editor de texto.
9. Copie o seguinte no arquivo `App.java`. Substitua *`your-neptune-endpoint`* pelo endereço da instância de banco de dados do Neptune.

> **Note**
>
> Para obter informações sobre como localizar o nome do host da instância de banco de dados do Neptune, consulte a seção Conectar-se a endpoints do Amazon Neptune.

```
package com.amazonaws;

import org.eclipse.rdf4j.repository.Repository;
import org.eclipse.rdf4j.repository.http.HTTPRepository;
import org.eclipse.rdf4j.repository.sparql.SPARQLRepository;

import java.util.List;
import org.eclipse.rdf4j.RDF4JException;
import org.eclipse.rdf4j.repository.RepositoryConnection;
import org.eclipse.rdf4j.query.TupleQuery;
import org.eclipse.rdf4j.query.TupleQueryResult;
import org.eclipse.rdf4j.query.BindingSet;
import org.eclipse.rdf4j.query.QueryLanguage;
```

```
import org.eclipse.rdf4j.model.Value;

public class App
{
    public static void main( String[] args )
    {
        String sparqlEndpoint = "https://your-neptune-endpoint:port/sparql";
        Repository repo = new SPARQLRepository(sparqlEndpoint);
        repo.initialize();

        try (RepositoryConnection conn = repo.getConnection()) {
           String queryString = "SELECT ?s ?p ?o WHERE { ?s ?p ?o } limit 10";

           TupleQuery tupleQuery = conn.prepareTupleQuery(QueryLanguage.SPARQL,
 queryString);

           try (TupleQueryResult result = tupleQuery.evaluate()) {
              while (result.hasNext()) {  // iterate over the result
                   BindingSet bindingSet = result.next();

                   Value s = bindingSet.getValue("s");
                   Value p = bindingSet.getValue("p");
                   Value o = bindingSet.getValue("o");

                   System.out.print(s);
                   System.out.print("\t");
                   System.out.print(p);
                   System.out.print("\t");
                   System.out.println(o);
              }
           }
        }
    }
}
```

10. Use o comando do Maven a seguir para compilar e executar o exemplo:

```
mvn compile exec:java
```

O exemplo anterior retorna até 10 dos triplos (subject-predicate-object) no gráfico usando a consulta `?s ?p ?o` com um limite de 10. Para consultar outro elemento, substitua a consulta por outra consulta do SPARQL.

A iteração dos resultados no exemplo imprime o valor de cada variável retornada. O objeto `Value` é convertido em um `String` e, em seguida, é impresso. Se alterar a parte `SELECT` da consulta, você deverá modificar o código.

API HTTP do SPARQL

As solicitações HTTP do SPARQL são aceitas no seguinte endpoint: `https://`*`your-neptune-endpoint`*`:`*`port`*`/sparql`

Para obter mais informações sobre como se conectar ao Amazon Neptune com o SPARQL, consulte Acessar o grafo do Neptune com o SPARQL.

Para obter mais informações sobre o protocolo e a linguagem de consulta SPARQL, consulte o Protocolo SPARQL 1.1 e a especificação SPARQL 1.1 Query Language.

Os tópicos a seguir fornecem informações sobre formatos de serialização do RDF SPARQL e como usar a API HTTP do SPARQL com Neptune.

Sumário

Usar o endpoint REST HTTP para conectar-se a uma instância de banco de dados do Neptune

O Amazon Neptune fornece um endpoint HTTP para consultas do SPARQL. A interface REST é compatível com o SPARQL versão 1.1.

> **Important**
>
> Versão: 1.0.4.0 (12/10/2020) tornou o TLS 1.2 e o HTTPS obrigatórios para todas as conexões com o Amazon Neptune. Não é mais possível se conectar ao Neptune usando HTTP não seguro nem HTTPS com uma versão do TLS anterior à 1.2.

As instruções a seguir explicam como conectar-se ao endpoint do SPARQL usando o comando curl e conectando-se por meio de HTTPS e usando a sintaxe HTTP. Siga estas instruções em uma instância do Amazon EC2 na mesma nuvem privada virtual (VPC) que a instância de banco de dados do Neptune.

O endpoint HTTP para consultas do SPARQL a uma instância de banco de dados do Neptune é `https://your-neptune-endpoint:port/sparql`.

> **Note**
>
> Para obter informações sobre como localizar o nome do host da instância de banco de dados do Neptune, consulte a seção Conectar-se a endpoints do Amazon Neptune.

CONSULTA usando HTTP POST

O exemplo a seguir usa o curl para enviar uma **`QUERY`** do SPARQL por meio do HTTP POST.

```
curl -X POST --data-binary 'query=select ?s ?p ?o where {?s ?p ?o} limit 10'
 https://your-neptune-endpoint:port/sparql
```

O exemplo anterior retorna até 10 dos triplos (subject-predicate-object) no gráfico usando a consulta `?s ?p ?o` com um limite de 10. Para consultar outro elemento, substitua-a por outra consulta do SPARQL.

> **Note**
>
> O tipo de mídia MIME padrão de uma resposta é `application/sparql-results+json` para consultas `SELECT` e `ASK`.
> O tipo MIME padrão de uma resposta é `application/n-quads` para `CONSTRUCT` e consultas `DESCRIBE`.

Para obter uma lista dos tipos de mídia usados pelo Neptune para serialização, consulte Formatos de serialização do RDF usados pelo SPARQL no Neptune.

UPDATE usando HTTP POST

O exemplo a seguir usa o curl para enviar uma **`UPDATE`** do SPARQL por meio do HTTP POST.

```
curl -X POST --data-binary 'update=INSERT DATA { <https://test.com/s> <https://
test.com/p> <https://test.com/o> . }' https://your-neptune-endpoint:port/sparql
```

O exemplo anterior insere o seguinte triplo no gráfico padrão do SPARQL: `<https://test.com/s> <https://test.com/p> <https://test.com/o>`

Cabeçalhos finais HTTP opcionais para respostas SPARQL de várias partes

Note

Esse atributo está disponível a partir da versão 1.0.3.0 do mecanismo do Neptune.

A resposta HTTP às consultas e atualizações do SPARQL geralmente é retornada em mais de uma parte ou fragmento. Pode ser difícil diagnosticar uma falha ocorrida após uma consulta ou uma atualização começar a enviar esses fragmentos, especialmente porque o primeiro chega com um código de status HTTP de `200`.

A menos que você solicite explicitamente os cabeçalhos finais, o Neptune só relata essa falha anexando uma mensagem de erro ao corpo da mensagem, que geralmente está corrompida.

Para facilitar a detecção e o diagnóstico desse tipo de problema, é possível incluir um cabeçalho de trailers com codificação de transferência (TE) (`te: trailers`) na solicitação (consulte, por exemplo, a página MDN sobre cabeçalhos de solicitação TE). Isso fará com que o Neptune inclua dois novos campos de cabeçalho nos cabeçalhos finais dos blocos de resposta:

- `X-Neptune-Status`: contém o código de resposta seguido por um nome curto. Por exemplo, em caso de êxito, o cabeçalho final seria: `X-Neptune-Status: 200 OK`. Em caso de falha, o código de resposta seria um código de erro do mecanismo do Neptune, como `X-Neptune-Status: 500 TimeLimitExceededException`.

- `X-Neptune-Detail`: fica em branco para solicitações bem-sucedidas. No caso de erros, ele contém a mensagem de erro JSON. Como somente caracteres ASCII são permitidos nos valores do cabeçalho HTTP, a string JSON é codificada em URL. A mensagem de erro também ainda é anexada ao corpo da mensagem de resposta.

Tipos de mídia do RDF usados pelo SPARQL no Neptune

Os dados Resource Description Framework (RDF) podem ser serializados de muitas maneiras diferentes. A maioria delas pode ser consumida ou gerada por SPARQL:

Formatos de serialização do RDF usados pelo SPARQL no Neptune

- RDF/XML: serialização XML do RDF, definida em RDF 1.1 XML Syntax. Tipo de mídia: `application/rdf+xml`. Extensão de arquivo típica: `.rdf`.
- N-Triples: formato de texto simples com base em linha para codificar um grafo do RDF, definido em RDF 1.1 N-Triples. Tipo de mídia: `application/n-triples`, `text/turtle` ou `text/plain`. Extensão de arquivo típica: `.nt`.
- N-Quads: formato de texto simples com base em linha para codificar um grafo do RDF, definido em RDF 1.1 N-Quads. Ele é uma extensão do N-Triples. Tipo de mídia: `application/n-quads` ou `text/x-nquads` quando codificado com US-ASCII de 7 bits. Extensão de arquivo típica: `.nq`.
- Turtle: sintaxe textual para RDF definida em RDF 1.1 Turtle que permite a um grafo do RDF ser completamente escrito em um formato de texto compacto e natural, com abreviações para tipos de dados e padrões de uso comuns. Turtle oferece níveis de compatibilidade com o formato N-Triples, bem como a sintaxe do padrão triplo do SPARQL. Tipo de mídia: `text/turtle`Extensão de arquivo típica: `.ttl`.
- TriG: sintaxe textual para RDF definida em RDF 1.1 TriG que permite a um grafo do RDF ser completamente escrito em um formato de texto compacto e natural, com abreviações para tipos de dados e padrões de uso comuns. TriG é uma extensão do formato Turtle. Tipo de mídia: `application/trig`. Extensão de arquivo típica: `.trig`.
- N3 (Notation3): linguagem de lógica e asserção definida em Notation3 (N3)Notation3 (N3): A readable RDF syntax. N3 estende o modelo de dados RDF adicionando fórmulas (literais que são gráficos em si), variáveis, implicação lógica e predicados funcionais, e fornece uma sintaxe textual alternativa para RDF/XML. Tipo de mídia: `text/n3`. Extensão de arquivo típica: `.n3`.
- JSON-LD: formato de mensagens e serialização de dados definido em JSON-LD 1.0.Media type: `application/ld+json`. Extensão de arquivo típica: `.jsonld`.

- TriX: serialização de RDF em XML, definida em TriX: RDF Triples in XML. Tipo de mídia: `application/trix`. Extensão de arquivo típica: `.trix`.
- Resultados SPARQL JSON: serialização do RDF usando o SPARQL 1.1 Query Results JSON Format. Tipo de mídia: `application/sparql-results+json`. Extensão de arquivo típica: `.srj`.
- Formato binário RDF4J: formato binário para codificar dados do RDF, documentado em RDF4J Binary RDF Format. Tipo de mídia: `application/x-binary-rdf`.

Formatos de serialização de resultado do SPARQL usados pelo SPARQL no Neptune

- Resultados XML do SPARQL: formato XML para os formatos de resultados boolianos e de vinculação variável fornecidos pela linguagem de consulta SPARQL, definida em SPARQL Query Results XML Format (Second Edition). Tipo de mídia: `application/sparql-results+xml`. Extensão de arquivo típica: `.srx`.
- Resultados CSV e TSV do SPARQL: a utilização de valores separados por vírgula ou por tabulação para expressar os resultados de consulta do SPARQL de consultas `SELECT`, definido em SPARQL 1.1 Query Results CSV and TSV Formats. Tipo de mídia: `text/csv` para valores separados por vírgula e `text/tab-separated-values` para valores separados por tabulação. Extensões de arquivo típicas: `.csv` para valores separados por vírgula e `.tsv` para valores separados por tabulação.
- Tabela de resultados binários: formato binário para codificar a saída de consultas do SPARQL. Tipo de mídia: `application/x-binary-rdf-results-table`.
- Resultados SPARQL JSON: serialização do RDF usando o SPARQL 1.1 Query Results JSON Format. Tipo de mídia: `application/sparql-results+json`.

Tipos de mídia que o Neptune pode usar para importar dados do RDF

Tipos de mídia compatíveis com o carregador em massa do Neptune

- N-Triples
- N-Quads
- RDF/XML
- Turtle

Tipos de mídia que SPARQL UPDATE LOAD pode importar

- N-Triples
- N-Quads
- RDF/XML
- Turtle
- TriG
- N3
- JSON-LD

Tipos de mídia que o Neptune pode usar para exportar resultados de consulta

Para especificar o formato de saída para uma resposta de consulta SPARQL, envie um cabeçalho "`Accept: media-type`" com a solicitação de consulta. Por exemplo: .

```
curl -H "Accept: application/nquads" ...
```

Tipos de mídia do RDF que o SPARQL SELECT pode gerar do Neptune

- Resultados JSON SPARQL (este é o padrão)
- Resultados XML SPARQL
- Tabela de resultados binários (tipo de mídia: `application/x-binary-rdf-results-table`)
- CSV (valores separados por vírgula)
- TSB (valores separados por tabulação)

Tipos de mídia do RDF que o SPARQL ASK pode gerar do Neptune

- Resultados JSON SPARQL (este é o padrão)
- Resultados XML SPARQL
- Booleano (tipo de mídia: `text/boolean`, o que significa "verdadeiro" ou "falso")

Tipos de mídia do RDF que o SPARQL CONSTRUCT pode gerar do Neptune

- N-Quads (este é o padrão)

- RDF/XML
- JSON-LD
- N-Triples
- Turtle
- N3
- TriX
- TriG
- Resultados JSON SPARQL
- Formato RDF binário RDF4J

Tipos de mídia do RDF que o SPARQL DESCRIBE pode gerar do Neptune

- N-Quads (este é o padrão)
- RDF/XML
- JSON-LD
- N-Triples
- Turtle
- N3
- TriX
- TriG
- Resultados JSON SPARQL
- Formato RDF binário RDF4J

Usar SPARQL UPDATE LOAD para importar dados para o Neptune

A sintaxe do comando SPARQL UPDATE LOAD é especificada na recomendação de atualização do SPARQL 1.1:

```
LOAD SILENT (URL of data to be loaded) INTO GRAPH (named graph into which to load the
 data)
```

- **`SILENT`**: (opcional) faz com que a operação seja bem-sucedida mesmo que tenha ocorrido um erro durante o processamento.

Isso pode ser útil quando uma única transação contém várias declarações, como "`LOAD ...; LOAD ...; UNLOAD ...; LOAD ...;`" e você deseja que a transação seja concluída mesmo que alguns dos dados remotos não possam ser processados.

- *`URL dos dados a serem carregados`*: (obrigatório) especifica um arquivo de dados remoto que contém dados a serem carregados em um grafo.

  O arquivo remoto deve ter uma das seguintes extensões:

  - `.nt` para NTriples.
  - `.nq` para NQuads.
  - `.trig` para Trig.
  - `.rdf` para RDF/XML.
  - `.ttl` para Turtle.
  - `.n3` para N3.
  - `.jsonld` para JSON-LD.
- **`INTO GRAPH`***`(gráfico nomeado no qual carregar os dados)`*: (opcional) especifica o grafo no qual os dados devem ser carregados.

  O Neptune associa cada triplo a um grafo nomeado. É possível especificar o grafo nomeado padrão usando o URI do grafo nomeado de fallback, `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`, da seguinte forma:

```
INTO GRAPH <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph>
```

> **Note**
>
> Quando você precisar carregar muitos dados, recomendamos usar o carregador em massa do Neptune em vez de UPDATE LOAD. Para obter mais informações sobre o carregador em massa, consulte Usar o carregador em massa do Amazon Neptune para ingerir dados.

É possível usar `SPARQL UPDATE LOAD` para carregar dados diretamente do Amazon S3 ou de arquivos obtidos de um servidor da web hospedado automaticamente. Os recursos a serem carregados devem residir na mesma região que o servidor do Neptune, e o endpoint para os

recursos deve ter permissão na VPC. Para obter informações sobre como criar um endpoint do Amazon S3, consulte Criar o endpoint da VPC do Amazon S3.

Todos os URIs `SPARQL UPDATE LOAD` devem começar com `https://`. Isso inclui URLs do Amazon S3.

Ao contrário do carregador em massa do Neptune, uma chamada para `SPARQL UPDATE LOAD` é totalmente transacional.

Carregar arquivos diretamente do Amazon S3 no Neptune usando SPARQL UPDATE LOAD

Como o Neptune não permite que você transmita um perfil do IAM para o Amazon S3 ao usar SPARQL UPDATE LOAD, o bucket do Amazon S3 em questão deve ser público ou você deve usar um URL do Amazon S3 pré-assinado na consulta LOAD.

Para gerar uma URL pré-assinada para um arquivo Amazon S3, você pode usar AWS CLI um comando como este:

```
aws s3 presign --expires-in (number of seconds) s3://(bucket name)/(path to file of
 data to load)
```

Depois, é possível usar o URL pré-assinado resultante no comando `LOAD`:

```
curl https://(a Neptune endpoint URL):8182/sparql \
  --data-urlencode 'update=load (pre-signed URL of the remote Amazon S3 file of data to
 be loaded) \
                           into graph (named graph)'
```

Para obter mais informações, consulte Autenticação de solicitações: usando parâmetros de consulta. A documentação do Boto3 mostra como usar um script Python para gerar um URL pré-assinado.

Além disso, o tipo de conteúdo dos arquivos a serem carregados deve ser definido corretamente.

1. Defina o tipo de conteúdo de arquivos ao fazer upload deles no Amazon S3 usando o parâmetro `-metadata`, como:

   ```
   aws s3 cp test.nt s3://bucket-name/my-plain-text-input/test.nt --metadata Content-
   Type=text/plain
   aws s3 cp test.rdf s3://bucket-name/my-rdf-input/test.rdf --metadata Content-
   Type=application/rdf+xml
   ```

2. Confirme se as informações de tipo de mídia estão realmente presentes. Execute:

   ```
   curl -v bucket-name/folder-name
   ```

   A saída desse comando deve mostrar as informações de tipo de mídia que você define ao carregar os arquivos.

3. Depois, você pode usar o comando `SPARQL UPDATE LOAD` para importar esses arquivos para o Neptune:

   ```
   curl https://your-neptune-endpoint:port/sparql \
     -d "update=LOAD <https://s3.amazonaws.com/bucket-name/my-rdf-input/test.rdf>"
   ```

As etapas acima funcionam apenas para um bucket público do Amazon S3 ou para um bucket que você acessa usando um URL do Amazon S3 pré-assinado na consulta LOAD.

Você também pode configurar um servidor de proxy web para carregar a partir de um bucket privado do Amazon S3, conforme mostrado abaixo:

Usar um servidor da web para carregar arquivos no Neptune com SPARQL UPDATE LOAD

1. Instale um servidor da web em uma máquina em execução na VPC que está hospedando Neptune e os arquivos a serem carregados. Por exemplo, usando o Amazon Linux, você pode instalar o Apache da seguinte forma:

   ```
   sudo yum install httpd mod_ssl
   sudo /usr/sbin/apachectl start
   ```

2. Defina os tipos MIME do conteúdo do arquivo RDF que você carregar. O SPARQL usa o cabeçalho `Content-type` enviado pelo servidor da Web para determinar o formato de entrada do conteúdo, portanto, é necessário definir tipos MIME relevantes para o servidor da Web.

   Por exemplo, suponha que você use as seguintes extensões de arquivo para identificar formatos de arquivo:

   - `.nt` para NTriples.
   - `.nq` para NQuads.
   - `.trig` para Trig.
   - `.rdf` para RDF/XML.

- `.ttl` para Turtle.
- `.n3` para N3.
- `.jsonld` para JSON-LD.

Se você estiver usando o Apache 2 como servidor da Web, edite o arquivo `/etc/mime.types` e adicione os seguintes tipos:

```
text/plain nt
application/n-quads nq
application/trig trig
application/rdf+xml rdf
application/x-turtle ttl
text/rdf+n3 n3
application/ld+json jsonld
```

3. Confirme se o mapeamento do tipo MIME funciona. Quando você tiver o servidor da Web em execução e hospedando arquivos RDF nos formatos de sua escolha, poderá testar a configuração enviando uma solicitação para o servidor da Web do seu host local.

   Por exemplo, você pode enviar uma solicitação como esta:

   ```
   curl -v http://localhost:80/test.rdf
   ```

   Em seguida, na saída detalhada de `curl`, você deve ver uma linha como:

   ```
   Content-Type: application/rdf+xml
   ```

   Isso mostra que o mapeamento de tipo de conteúdo foi definido com êxito.

4. Agora você está com tudo pronto para carregar dados usando o comando SPARQL UDPATE:

   ```
   curl https://your-neptune-endpoint:port/sparql \
       -d "update=LOAD <http://web_server_private_ip:80/test.rdf>"
   ```

> **Note**
>
> O uso de `SPARQL UPDATE LOAD` pode acionar um tempo limite no servidor da Web quando o arquivo de origem que está sendo carregado é grande. O Neptune processa os dados

do arquivo à medida que são transmitidos, e, para um arquivo grande, isso pode demorar mais do que o tempo limite configurado no servidor. Isso, por sua vez, pode fazer com que o servidor feche a conexão, o que pode gerar a seguinte mensagem de erro quando o Neptune encontrar um EOF inesperado no fluxo:

```
{
  "detailedMessage":"Invalid syntax in the specified file",
  "code":"InvalidParameterException"
}
```

Se receber essa mensagem e não acreditar que o arquivo de origem contenha uma sintaxe inválida, tente aumentar as definições de tempo limite no servidor da Web. Também é possível diagnosticar o problema ativando logs de depuração no servidor e procurando tempos limite.

Usar o SPARQL UPDATE UNLOAD para excluir dados do Neptune

O Neptune também oferece uma operação SPARQL personalizada, `UNLOAD`, para remover dados especificados em uma fonte remota. `UNLOAD` pode ser considerado uma contrapartida da operação `LOAD`. A sintaxe é:

> **Note**
>
> Esse atributo está disponível a partir da versão 1.0.4.1 do mecanismo do Neptune.

```
UNLOAD SILENT (URL of the remote data to be unloaded) FROM GRAPH (named graph from
 which to remove the data)
```

- **`SILENT`**: (opcional) faz com que a operação seja bem-sucedida mesmo que tenha ocorrido um erro durante o processamento dos dados.

  Isso pode ser útil quando uma única transação contém várias declarações, como "`LOAD ...; LOAD ...; UNLOAD ...; LOAD ...;`" e você deseja que a transação seja concluída mesmo que alguns dos dados remotos não possam ser processados.
- *`URL dos dados remotos a serem descarregados`*: (obrigatório) especifica um arquivo de dados remoto que contém dados a serem descarregados em um grafo.

O arquivo remoto deve ter uma das seguintes extensões (são os mesmos formatos compatíveis com UPDATE-LOAD):

- `.nt` para NTriples.
- `.nq` para NQuads.
- `.trig` para Trig.
- `.rdf` para RDF/XML.
- `.ttl` para Turtle.
- `.n3` para N3.
- `.jsonld` para JSON-LD.

Todos os dados que esse arquivo contém serão removidos do cluster de banco de dados pela operação `UNLOAD`.

Qualquer autenticação do Amazon S3 deve ser incluída no URL para que os dados sejam descarregados. É possível pré-assinar um arquivo do Amazon S3 e depois usar o URL resultante para acessá-lo com segurança. Por exemplo: .

```
aws s3 presign --expires-in (number of seconds) s3://(bucket name)/(path to file of
 data to unload)
```

Em seguida:

```
curl https://(a Neptune endpoint URL):8182/sparql \
  --data-urlencode 'update=unload (pre-signed URL of the remote Amazon S3 data to be
 unloaded) \
                           from graph (named graph)'
```

Para obter mais informações, consulte Autenticação de solicitações: usando parâmetros de consulta.

- **`FROM GRAPH`** *`(gráfico nomeado do qual remover os dados)`*: (opcional) especifica o grafo nomeado do qual os dados remotos devem ser descarregados.

  O Neptune associa cada triplo a um grafo nomeado. É possível especificar o grafo nomeado padrão usando o URI do grafo nomeado de fallback, `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`, da seguinte forma:

```
FROM GRAPH <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph>
```

Da mesma forma que LOAD corresponde a `INSERT DATA { (inline data) }`, UNLOAD corresponde a `DELETE DATA { (inline data) }`. Como `DELETE DATA`, UNLOAD não funciona em dados com nós em branco.

Por exemplo, se um servidor web local fornecer um arquivo denominado `data.nt` que contenha estes dois triplos:

```
<http://example.org/resource#a> <http://example.org/resource#p> <http://example.org/
resource#b> .
<http://example.org/resource#a> <http://example.org/resource#p> <http://example.org/
resource#c> .
```

O seguinte comando UNLOAD excluirá esses dois triplos do grafo nomeado, `<http://example.org/graph1>`:

```
UNLOAD <http://localhost:80/data.nt> FROM GRAPH <http://example.org/graph1>
```

Isso terá o mesmo efeito que usar o seguinte comando `DELETE DATA`:

```
DELETE DATA {
  GRAPH <http://example.org/graph1> {
    <http://example.org/resource#a> <http://example.org/resource#p> <http://
example.org/resource#b> .
    <http://example.org/resource#a> <http://example.org/resource#p> <http://
example.org/resource#c> .
  }
}
```

Exceções lançadas pelo comando **`UNLOAD`**.

- **`InvalidParameterException`**: havia nós em branco nos dados. Status HTTP: 400 Solicitação inválida.

  Mensagem: `Blank nodes are not allowed for UNLOAD`

- **`InvalidParameterException`**: havia uma sintaxe rompida nos dados. Status HTTP: 400 Solicitação inválida.

  Mensagem: `Invalid syntax in the specified file.`

- **`UnloadUrlAccessDeniedException`** : acesso negado. Status HTTP: 400 Solicitação inválida.

  Mensagem: `Update failure: Endpoint` *`(Neptune endpoint)`* `reported access denied error. Please verify access.`

- **`BadRequestException`** : os dados remotos não podem ser recuperados. Status HTTP: 400 Solicitação inválida.

  Mensagem: (depende da resposta HTTP).

Dicas de consulta do SPARQL

É possível usar as dicas de consulta para especificar estratégias de otimização e avaliação para uma determinada consulta SPARQL no Amazon Neptune.

As dicas de consulta são expressas usando padrões triplos adicionais incorporados na consulta SPARQL com as seguintes partes:

```
scope hint value
```

- escopo: determina a parte da consulta à qual a dica de consulta se aplica, como um grupo específico na consulta ou a consulta completa.
- dica: identifica o tipo de dica a ser aplicada.
- value: determina o comportamento do aspecto do sistema em consideração.

As dicas de consulta e os escopos são expostos como termos predefinidos no namespace `http://aws.amazon.com/neptune/vocab/v01/QueryHints#` do Amazon Neptune. Os exemplos desta seção incluem o namespace como um prefixo `hint` que é definido e incluído na consulta:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
```

O exemplo a seguir mostra como incluir uma dica `joinOrder` em uma consulta `SELECT`:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT ... {
 hint:Query hint:joinOrder "Ordered" .
 ...
}
```

A consulta anterior instrui o mecanismo do Neptune a avaliar junções na consulta em determinada ordem e desabilita qualquer reordenação automática.

Considere o seguinte ao usar as dicas de consulta:

- É possível combinar dicas de consulta diferentes em uma única consulta. Por exemplo, é possível usar a dica de consulta `bottomUp` para anotar uma subconsulta para uma avaliação de baixo para cima e uma dica de consulta `joinOrder` para corrigir a ordem da junção dentro da subconsulta.
- É possível usar a mesma dica de consulta várias vezes, em diferentes escopos não sobrepostos
- Dicas de consultas são dicas. Embora o mecanismo de consulta geralmente tenha por objetivo considerar determinadas dicas de consulta, ele também pode ignorá-las.
- As dicas de consulta são de preservação semântica. Adicionar uma dica de consulta não altera a saída da consulta (exceto pela possível ordem dos resultados quando nenhuma garantia de ordenação é oferecida, ou seja, quando a ordem dos resultados não é aplicada explicitamente usando ORDER BY).

As seções a seguir fornecem mais informações sobre as dicas de consulta disponíveis e o uso no Neptune.

Tópicos

Escopo de dicas de consulta do SPARQL no Neptune

A tabela a seguir mostra os escopos disponíveis, as dicas associadas e as descrições de dicas de consulta do SPARQL no Amazon Neptune. O prefixo `hint` nessas entradas representa o namespace do Neptune para dicas:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
```

Escopo	Dica compatível	Descrição
`hint:Query`	joinOrder	A dica de consulta se aplica à consulta inteira.
`hint:Query`	queryTimeout	O valor de tempo limite se aplica a toda a consulta.
`hint:Query`	rangeSafe	A promoção de tipo está desabilitada para toda a consulta.
`hint:Query`	queryId	O valor de ID de consulta se aplica a toda a consulta.
`hint:Query`	useDFE	O uso do DFE está habilitado (ou desabilitado) para toda a consulta.
`hint:Group`	joinOrder	A dica de consulta se aplica aos elementos de nível superior no grupo especificado, mas não a elementos aninhados (como as subconsultas) ou elementos pai.
`hint:SubQuery`	evaluationStrategy	A dica é especificada e aplicada a uma subconsul ta SELECT aninhada. A subconsulta é avaliada de

Escopo	Dica compatível	Descrição
		maneira independente, sem considerar as soluções calculadas antes da subconsulta.

A dica de consulta **`joinOrder`** SPARQL

Quando você envia uma consulta do SPARQL, o mecanismo de consulta do Amazon Neptune investiga a estrutura da consulta. Ele reordena partes da consulta e tenta minimizar a quantidade de trabalho necessária para a avaliação e o tempo de resposta da consulta.

Por exemplo, uma sequência de padrões triplos conectados normalmente não é avaliada na ordem indicada. Ela é reordenada usando heurística e estatísticas, como a seletividade dos padrões individuais e como eles estão conectados por meio de variáveis compartilhadas. Além disso, se sua consulta contém padrões mais complexos, como subconsultas, FILTROs, ou blocos MINUS ou OPTIONAL complexos, o mecanismo de consulta do Neptune reordena-os quando possível, mirando uma ordem de avaliação eficiente.

Para consultas mais complexas, a ordem que o Neptune escolhe para avaliar a consulta pode não ser sempre a ideal. Por exemplo, o Neptune pode perder características específicas de dados da instância (como atingir nós avançados no grafo) que surgem durante a avaliação da consulta.

Se você sabe exatamente as características dos dados e deseja ditar manualmente a ordem de execução da consulta, use a dica de consulta `joinOrder` do Neptune para especificar que a consulta deve ser avaliada na ordem indicada.

Sintaxe de dica **`joinOrder`** SPARQL

A dica de consulta `joinOrder` é especificada como um padrão triplo incluído em uma consulta SPARQL.

Para fins de clareza, a seguinte sintaxe usa um prefixo `hint` definido e incluído na consulta para especificar o namespace de dica de consulta do Neptune:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
scope hint:joinOrder "Ordered" .
```

Escopos disponíveis

- `hint:Query`
- `hint:Group`

Para obter mais informações sobre escopos de dica de consulta, consulte Escopo de dicas de consulta do SPARQL no Neptune.

Exemplo de dica **`joinOrder`** SPARQL

Esta seção mostra uma consulta gravada com e sem a dica de consulta `joinOrder` e otimizações relacionadas.

Para este exemplo, suponha que o conjunto de dados contenha o seguinte:

- Uma única pessoa chamada `John` que curte (`:likes`) 1.000 pessoas, inclusive `Jane`.
- Uma única pessoa chamada `Jane` que curte (`:likes`) 10 pessoas, inclusive `John`.

Nenhuma dica de consulta

A seguinte consulta SPARQL extrai todos os pares de pessoas chamados `John` e `Jane` que curtem um ao outro de um conjunto de dados de redes sociais:

```
PREFIX : <https://example.com/>
SELECT ?john ?jane {
  ?person1 :name "Jane" .
  ?person1 :likes ?person2 .
  ?person2 :name "John" .
  ?person2 :likes ?person1 .
}
```

O mecanismo de consulta do Neptune pode avaliar as declarações em uma ordem diferente da escrita. Por exemplo, ele pode escolher avaliar na seguinte ordem:

1. Encontrar todas as pessoas chamadas `John`.
2. Encontrar todas as pessoas conectadas a `John` por uma borda `:likes`.
3. Filtrar esse conjunto por pessoas chamadas `Jane`.
4. Filtrar esse conjunto por aqueles indivíduos conectados a `John` por uma borda `:likes`.

De acordo com o conjunto de dados, avaliar nesta ordem resulta em 1.000 entidades sendo extraídas na segunda etapa. A terceira etapa restringe isso ao único nó, `Jane`. A etapa final determina que `Jane` também curte (`:likes`) o nó `John`.

Dica de consulta

Seria favorável começar com o nó `Jane` porque ela tem apenas 10 bordas `:likes` de saída. Isso reduz a quantidade de trabalho durante a avaliação da consulta, evitando a extração de 1.000 entidades durante a segunda etapa.

O exemplo a seguir usa a consulta joinOrder para garantir que o nó `Jane` e suas bordas de saída sejam processados primeiro ao desabilitar toda a reordenação automática de junção da consulta:

```
PREFIX : <https://example.com/>
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT ?john ?jane {
  hint:Query hint:joinOrder "Ordered" .
  ?person1 :name "Jane" .
  ?person1 :likes ?person2 .
  ?person2 :name "John" .
  ?person2 :likes ?person1 .
}
```

Um cenário do mundo real aplicável pode ser um aplicativo de rede social no qual as pessoas na rede são classificadas tanto como influenciadoras com muitas conexões ou como usuários normais com poucas conexões. Em um cenário como esse, você pode garantir que o usuário normal (`Jane`) seja processado antes do influenciador (`John`) em uma consulta como o exemplo anterior.

Dica de consulta e reordenação

Você pode levar este exemplo uma etapa adiante. Se você sabe que o atributo `:name` é exclusivo para um único nó, pode acelerar a consulta ao reorganizar e usar a dica de consulta `joinOrder`. Essa etapa garante que os nós exclusivos sejam extraídos primeiro.

```
PREFIX : <https://example.com/>
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT ?john ?jane {
  hint:Query hint:joinOrder "Ordered" .
  ?person1 :name "Jane" .
  ?person2 :name "John" .
  ?person1 :likes ?person2 .
```

```
  ?person2 :likes ?person1 .
}
```

Nesse caso, você pode reduzir a consulta às seguintes ações individuais em cada etapa:

1. Encontrar o nó de uma única pessoa com `:name Jane`.
2. Encontrar o nó de uma única pessoa com `:name John`.
3. Verificar se o primeiro nó está conectado ao segundo com uma borda `:likes`.
4. Verificar se o segundo nó está conectado ao primeiro com uma borda `:likes`.

> **Important**
>
> Se você escolher a ordem errada, a dica de consulta `joinOrder` poderá resultar em quedas de desempenho significativas. Por exemplo, o exemplo anterior seria ineficaz se os atributos `:name` não fossem exclusivos. Se todos os 100 nós fossem nomeados `Jane` e todos os 1.000 nós fossem nomeados `John`, a consulta terminaria verificando 1.000 * 100 (100.000) pares para bordas `:likes`.

A dica de consulta **`evaluationStrategy`** SPARQL

A dica de consulta `evaluationStrategy` informa ao mecanismo de consulta do Amazon Neptune que o fragmento de consulta anotado deve ser avaliado de baixo para cima, como uma unidade independente. Isso significa que nenhuma solução das etapas anteriores da avaliação é usada para computar o fragmento de consulta. O fragmento de consulta é avaliado como uma unidade autônoma, e suas soluções produzidas são unidas ao restante da consulta após serem computadas.

Usar a dica de consulta `evaluationStrategy` implica em um bloqueio (sem pipeline) do plano de consulta, o que significa que as soluções do fragmento anotado com a dica de consulta são materializadas e armazenadas em buffer na memória principal. Usar essa dica de consulta pode aumentar significativamente a quantidade de memória principal necessária para avaliar a consulta, especialmente se o fragmento de consulta anotado calcula um grande número de resultados.

Sintaxe de dica **`evaluationStrategy`** SPARQL

A dica de consulta `evaluationStrategy` é especificada como um padrão triplo incluído em uma consulta SPARQL.

Para fins de clareza, a seguinte sintaxe usa um prefixo `hint` definido e incluído na consulta para especificar o namespace de dica de consulta do Neptune:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
hint:SubQuery hint:evaluationStrategy "BottomUp" .
```

Escopos disponíveis

- `hint:SubQuery`

> **Note**
>
> Essa dica de consulta é compatível apenas com subconsultas aninhadas.

Para obter mais informações sobre escopos de dica de consulta, consulte Escopo de dicas de consulta do SPARQL no Neptune.

Exemplo de dica **`evaluationStrategy`** SPARQL

Esta seção mostra uma consulta gravada com e sem a dica de consulta `evaluationStrategy` e otimizações relacionadas.

Para este exemplo, suponha que o dataset tenha as seguintes características:

- Ele contém 1.000 pontos rotulados `:connectedTo`.
- Cada nó `component` está conectado a uma média de 100 outros nós `component`.
- O número típico de conexões cíclicas de quatro saltos entre nós é em torno de 100.

Nenhuma dica de consulta

A seguinte consulta SPARQL extrai todos os nós `component` que estão ciclicamente conectados entre si por meio de quatro saltos:

```
PREFIX : <https://example.com/>
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT * {
```

```
  ?component1 :connectedTo ?component2 .
  ?component2 :connectedTo ?component3 .
  ?component3 :connectedTo ?component4 .
  ?component4 :connectedTo ?component1 .
}
```

A abordagem do mecanismo de consulta do Neptune é avaliar essa consulta usando as seguintes etapas:

- Extrair todos os 1.000 pontos `connectedTo` no gráfico.
- Expandir em 100x (o número de pontos de saída `connectedTo` de component2).

  Resultados intermediários: 100.000 nós.
- Expandir em 100x (o número de pontos de saída `connectedTo` de component3).

  Resultados intermediários: 10.000.000 nós.
- Examinar os 10.000.000 nós para o fechamento do ciclo.

Isso resulta em um plano de consulta de streaming, que tem uma quantidade constante de memória principal.

Dica de consulta e subconsultas

Talvez você queira trocar o espaço da memória principal para computação acelerada. Ao reescrever a consulta usando uma dica de consulta `evaluationStrategy`, você pode forçar o mecanismo a calcular uma junção entre dois subconjuntos menores e materializados.

```
PREFIX : <https://example.com/>
          PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT * {
  {
    SELECT * WHERE {
      hint:SubQuery hint:evaluationStrategy "BottomUp" .
      ?component1 :connectedTo ?component2 .
      ?component2 :connectedTo ?component3 .
    }
  }
  {
    SELECT * WHERE {
      hint:SubQuery hint:evaluationStrategy "BottomUp" .
```

```
      ?component3 :connectedTo ?component4 .
      ?component4 :connectedTo ?component1 .
    }
  }
}
```

Em vez de avaliar os padrões triplos em sequência enquanto usa interativamente os resultados dos padrões interativos triplos anteriores como entrada para os próximos padrões, a dica `evaluationStrategy` faz com que duas subconsultas sejam avaliadas de forma independente. Ambas as subconsultas produzem 100.000 nós para os resultados intermediários, que são, por sua vez, unidos para formar a saída final.

Especificamente, quando você executar o Neptune nos tipos de instância maiores, armazenar temporariamente esses dois subconjuntos de cem mil nós na memória principal aumentará o uso de memória em troca de acelerar significativamente a avaliação.

A dica de consulta **`queryTimeout`** SPARQL

A dica de consulta `queryTimeout` especifica um tempo limite menor do que o valor `neptune_query_timeout` definido no grupo de parâmetros do banco de dados.

Se a consulta for encerrada como resultado dessa dica, um `TimeLimitExceededException` será lançado com uma mensagem `Operation terminated (deadline exceeded)`.

Sintaxe de dica **`queryTimeout`** SPARQL

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT ... WHERE {
    hint:Query hint:queryTimeout 10 .
    # OR
    hint:Query hint:queryTimeout "10" .
    # OR
    hint:Query hint:queryTimeout "10"^^xsd:integer .
 ...
}
```

O valor de tempo limite é expresso em milissegundos.

O valor de tempo limite deve ser menor que o valor `neptune_query_timeout` definido no grupo de parâmetros de banco de dados. Caso contrário, uma exceção `MalformedQueryException` será

gerada com uma mensagem `Malformed query: Query hint 'queryTimeout' must be less than neptune_query_timeout DB Parameter Group`.

A dica de consulta `queryTimeout` deve ser especificada na cláusula `WHERE` da consulta principal, ou na cláusula `WHERE` de uma das subconsultas, conforme mostrado no exemplo a seguir.

Ela deve ser definida apenas uma vez em todas as consultas/subconsultas e seções de atualização SPARQL (como INSERT e DELETE). Caso contrário, uma exceção `MalformedQueryException` será gerada com uma mensagem `Malformed query: Query hint 'queryTimeout' must be set only once`.

Escopos disponíveis

A dica `queryTimeout` pode ser aplicada a consultas e atualizações SPARQL.

- Em uma consulta SPARQL, ela pode aparecer na cláusula WHERE da consulta principal ou em uma subconsulta.
- Em uma atualização SPARQL, ela pode ser definida na cláusula INSERT, DELETE ou WHERE. Se houver várias cláusulas de atualização, ela só poderá ser definida em uma delas.

Para obter mais informações sobre escopos de dica de consulta, consulte Escopo de dicas de consulta do SPARQL no Neptune.

Exemplo de dica **`queryTimeout`** SPARQL

Veja um exemplo de uso de `hint:queryTimeout` na cláusula principal `WHERE` de uma consulta `UPDATE`:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
INSERT {
    ?s ?p ?o
} WHERE {
    hint:Query hint:queryTimeout 100 .
    ?s ?p ?o .
}
```

Aqui, `hint:queryTimeout` está na cláusula `WHERE` de uma subconsulta:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
```

```
SELECT * {
   ?s ?p ?o .
   {
      SELECT ?s WHERE {
         hint:Query hint:queryTimeout 100 .
         ?s ?p1 ?o1 .
      }
   }
}
```

A dica de consulta **`rangeSafe`** SPARQL

Use essa dica de consulta para desativar a promoção de tipo para uma consulta SPARQL.

Quando você envia uma consulta do SPARQL que inclua um `FILTER` por um intervalo ou um valor numérico, o mecanismo de consulta do Neptune normalmente deve usar promoção de tipo ao executar a consulta. Isso significa que ele precisa examinar valores de todos os tipos que possam conter o valor que você está filtrando.

Por exemplo, se você estiver filtrando valores iguais a 55, o mecanismo deverá procurar números inteiros iguais a 55, números inteiros longos iguais a 55L, flutuantes iguais a 55,0, etc. Cada promoção de tipo exige uma pesquisa adicional no armazenamento, o que pode fazer com que uma consulta aparentemente simples leve um tempo inesperadamente longo para ser concluída.

Muitas vezes, a promoção de tipo é desnecessária porque você sabe de antemão que só precisa encontrar valores de um tipo específico. Nesse caso, você pode acelerar suas consultas drasticamente usando a dica de consulta `rangeSafe` para desativar a promoção de tipos.

Sintaxe de dica **`rangeSafe`** SPARQL

A dica de consulta `rangeSafe` assume o valor `true` para desativar a promoção de tipo. Ela também aceita o valor `false` (o padrão).

Exemplo. O seguinte exemplo mostra como desativar a promoção de tipo ao filtrar por um valor inteiro o maior que 1:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT * {
   ?s ?p ?o .
   hint:Prior hint:rangeSafe 'true' .
   FILTER (?o > '1'^^<http://www.w3.org/2001/XMLSchema#int>)
```

A dica de consulta **`queryId`** SPARQL

Use esta dica de consulta para atribuir seu próprio valor de queryId a uma consulta SPARQL.

Exemplo:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT * WHERE {
  hint:Query hint:queryId "4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47"
  {?s ?p ?o}}
```

O valor atribuído deve ser exclusivo em todas as consultas no banco de dados do Neptune.

A dica de consulta **`useDFE`** SPARQL

Use essa dica de consulta para permitir o uso do DFE para executar a consulta. Por padrão, o Neptune não usa o DFE sem que essa dica de consulta esteja definida como `true`, porque o parâmetro de instância neptune_dfe_query_engine tem como padrão `viaQueryHint`. Se você definir esse parâmetro de instância como `enabled`, o mecanismo do DFE será usado para todas as consultas, exceto aquelas com a dica de consulta `useDFE` definida como `false`.

Exemplo de habilitação do uso do DFE para uma consulta:

```
PREFIX : <https://example.com/>
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>

SELECT ?john ?jane
{
  hint:Query hint:useDFE true .
  ?person1 :name "Jane" .
  ?person1 :likes ?person2 .
  ?person2 :name "John" .
  ?person2 :likes ?person1 .
}
```

Dicas de consulta SPARQL usadas com DESCRIBE

Uma consulta `DESCRIBE` SPARQL oferece um mecanismo flexível para solicitar descrições de recursos. No entanto, as especificações do SPARQL não definem a semântica precisa de `DESCRIBE`.

A partir da versão 1.2.0.2 do mecanismo, o Neptune é compatível com vários modos `DESCRIBE` e algoritmos diferentes que são adequados para situações diferentes.

Esse exemplo de conjunto de dados pode ajudar a ilustrar os diferentes modos:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix : <https://example.com/> .

:JaneDoe :firstName "Jane" .
:JaneDoe :knows :JohnDoe .
:JohnDoe :firstName "John" .
:JaneDoe :knows _:b1 .
_:b1 :knows :RichardRoe .

:RichardRoe :knows :JaneDoe .
:RichardRoe :firstName "Richard" .

_:s1 rdf:type rdf:Statement .
_:s1 rdf:subject :JaneDoe .
_:s1 rdf:predicate :knows .
_:s1 rdf:object :JohnDoe .
_:s1 :knowsFrom "Berlin" .

:ref_s2 rdf:type rdf:Statement .
:ref_s2 rdf:subject :JaneDoe .
:ref_s2 rdf:predicate :knows .
:ref_s2 rdf:object :JohnDoe .
:ref_s2 :knowsSince 1988 .
```

Os exemplos abaixo pressupõem que uma descrição do recurso `:JaneDoe` está sendo solicitada usando uma consulta SPARQL como esta:

```
DESCRIBE <https://example.com/JaneDoe>
```

A dica de consulta **`describeMode`** SPARQL

A dica de consulta `hint:describeMode` SPARQL é usada para selecionar um dos seguintes modos `DESCRIBE` SPARQL compatíveis com o Neptune:

O modo **`ForwardOneStep`** DESCRIBE

Você invoca o modo `ForwardOneStep` com a dica de consulta `describeMode` desta forma:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
DESCRIBE <https://example.com/JaneDoe>
{
  hint:Query hint:describeMode "ForwardOneStep"
}
```

O modo `ForwardOneStep` gera somente os atributos e os links diretos do recurso a ser descrito. No caso de exemplo, isso significa que ele gera os triplos que têm `:JaneDoe`, o recurso a ser descrito, como assunto:

```
:JaneDoe :firstName "Jane" .
:JaneDoe :knows :JohnDoe .
:JaneDoe :knows _:b301990159 .
```

Observe que a consulta DESCRIBE pode gerar triplos com nós em branco, como `_:b301990159`, que têm IDs diferentes a cada vez, em comparação ao conjunto de dados de entrada.

O modo **`SymmetricOneStep`** DESCRIBE

`SymmetricOneStep` é o modo DESCRIBE padrão caso você não forneça uma dica de consulta. Você também pode invocá-la explicitamente com uma dica de consulta `describeMode` da seguinte forma:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
DESCRIBE <https://example.com/JaneDoe>
{
  hint:Query hint:describeMode "SymmetricOneStep"
}
```

Na semântica `SymmetricOneStep`, `DESCRIBE` retorna os atributos, links diretos e links reversos do recurso a ser descrito:

```
:JaneDoe :firstName "Jane" .
:JaneDoe :knows :JohnDoe .
:JaneDoe :knows _:b318767375 .

_:b318767631 rdf:subject :JaneDoe .

:RichardRoe :knows :JaneDoe .
```

```
:ref_s2 rdf:subject :JaneDoe .
```

O modo Concise Bounded Description (**CBD**) DESCRIBE

O modo Concise Bounded Description (`CBD`) é invocado usando a dica de consulta `describeMode` da seguinte forma:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
DESCRIBE <https://example.com/JaneDoe>
{
  hint:Query hint:describeMode "CBD"
}
```

Na semântica `CBD`, `DESCRIBE` gera o modo Concise Bounded Description (conforme definido pelo W3C) do recurso a ser descrito:

```
:JaneDoe :firstName "Jane" .
:JaneDoe :knows :JohnDoe .
:JaneDoe :knows _:b285212943 .
_:b285212943 :knows :RichardRoe .

_:b285213199 rdf:subject :JaneDoe .
_:b285213199 rdf:type rdf:Statement .
_:b285213199 rdf:predicate :knows .
_:b285213199 rdf:object :JohnDoe .
_:b285213199 :knowsFrom "Berlin" .

:ref_s2 rdf:subject :JaneDoe .
```

O modo Concise Bounded Description de um recurso do RDF (ou seja, um nó em um grafo do RDF) é o menor subgrafo centralizado em torno desse nó que pode ficar independente. Na prática, isso significa que, se você pensar nesse grafo como uma árvore, com o nó designado como raiz, não haverá nós em branco (nós) como folhas dessa árvore. Como os nós não podem ser tratados externamente nem usados em consultas subsequentes, não basta navegar no grafo apenas para encontrar os próximos saltos únicos do nó atual. Você também precisa ir longe o suficiente para encontrar algo que possa ser usado em consultas subsequentes (ou seja, algo diferente de um bnode).

Calcular o CBD

Dado um nó específico (o nó inicial ou raiz) no grafo do RDF de origem, o CBD desse nó é calculado da seguinte forma:

1. Inclua no subgrafo todas as declarações no grafo de origem em que o assunto da declaração é o nó inicial.
2. Recursivamente, até o momento, para todas as declarações no subgrafo que têm um objeto de nó em branco, inclua no subgrafo todas as declarações no grafo de origem em que o assunto da declaração seja esse nó em branco e que ainda não tenham sido incluídas no subgrafo.
3. Recursivamente, até o momento, para todas as declarações incluídas no subgrafo, para todas as reificações dessas declarações no grafo de origem, inclua o CBD começando no nó `rdf:Statement` de cada retificação.

Isso gera um subgrafo em que os nós do objeto são referências ou literais do IRI, ou nós em branco que não servem como assunto de nenhuma declaração no grafo. Observe que o CBD não pode ser calculado usando uma única consulta SPARQL SELECT ou CONSTRUCT.

O modo Symmetric Concise Bounded Description (**`SCBD`**) DESCRIBE

O modo Symmetric Concise Bounded Description (`SCBD`) é invocado usando a dica de consulta `describeMode` da seguinte forma:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
DESCRIBE <https://example.com/JaneDoe>
{
  hint:Query hint:describeMode "SCBD"
}
```

Na semântica `SCBD`, `DESCRIBE` gera o modo Symmetric Concise Bounded Description do recurso (conforme definido pelo W3C em Describing Linked Datasets with the VoID Vocabulary:

```
:JaneDoe :firstName "Jane" .
:JaneDoe :knows :JohnDoe .
:JaneDoe :knows _:b335544591 .
_:b335544591 :knows :RichardRoe .

:RichardRoe :knows :JaneDoe .
```

```
_:b335544847 rdf:subject :JaneDoe .
_:b335544847 rdf:type rdf:Statement .
_:b335544847 rdf:predicate :knows .
_:b335544847 rdf:object :JohnDoe .
_:b335544847 :knowsFrom "Berlin" .

:ref_s2 rdf:subject :JaneDoe .
```

A vantagem do CBD e do SCBD em relação aos modos `ForwardOneStep` e `SymmetricOneStep` é que os nós em branco são sempre expandidos para incluir a respectiva representação. Isso pode ser uma vantagem importante porque não é possível consultar um nó em branco usando o SPARQL. Além disso, os modos CBD e SCBD também consideram as reificações.

Observe que a dica de consulta `describeMode` também pode fazer parte de uma cláusula `WHERE`:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
DESCRIBE ?s
WHERE {
  hint:Query hint:describeMode "CBD" .
  ?s rdf:type <https://example.com/Person>
}
```

A dica de consulta **`describeIterationLimit`** SPARQL

A dica de consulta `hint:describeIterationLimit` SPARQL fornece uma restrição opcional sobre o número máximo de expansões iterativas a serem executadas para algoritmos DESCRIBE iterativos, como CBD e SCBD.

Os limites de DESCRIBE são associados. Portanto, se o limite de iteração e o limite de declarações forem especificados, os dois limites deverão ser atendidos antes que a consulta DESCRIBE seja eliminada.

O padrão para esse valor é cinco. Você pode defini-lo como ZERO (0) para especificar SEM limite no número de expansões iterativas.

A dica de consulta **`describeStatementLimit`** SPARQL

A dica de consulta `hint:describeStatementLimit` SPARQL fornece uma restrição opcional sobre o número máximo de declarações que podem estar presentes em uma resposta de consulta DESCRIBE. Ele é aplicado apenas para algoritmos DESCRIBE iterativos, como CBD e SCBD.

Os limites de DESCRIBE são associados. Portanto, se o limite de iteração e o limite de declarações forem especificados, os dois limites deverão ser atendidos antes que a consulta DESCRIBE seja eliminada.

O padrão para esse valor é cinco mil. Você pode defini-lo como ZERO (0) para especificar SEM limite no número de declarações geradas.

Comportamento do SPARQL DESCRIBE em relação ao grafo padrão

O formato da consulta `DESCRIBE` SPARQL permite que você recupere informações sobre recursos sem conhecer a estrutura dos dados e sem precisar compor uma consulta. A forma como essas informações são reunidas é deixada para a implementação do SPARQL. O Neptune fornece várias dicas de consulta que invocam diferentes modos e algoritmos para `DESCRIBE` usar.

Na implementação do Neptune, independentemente do modo, `DESCRIBE` usa apenas dados presentes no grafo padrão do SPARQL. Isso é consistente com a forma como o SPARQL trata os conjuntos de dados (consulte Specifying RDF Datasets na especificação do SPARQL).

No Neptune, o grafo padrão contém todos os triplos exclusivos na união de todos os grafos nomeados no banco de dados, a menos que grafos nomeados específicos sejam especificados usando cláusulas `FROM` e/ou `FROM NAMED`. Todos os dados do RDF no Neptune são armazenados em um grafo nomeado. Se um triplo for inserido sem um contexto de grafo nomeado, o Neptune o armazenará em um grafo nomeado designado `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`.

Quando um ou mais grafos nomeados são especificados usando a cláusula `FROM`, o grafo padrão é a união de todos os triplos exclusivos nesses grafos nomeados. Se não houver nenhuma cláusula `FROM` e houver uma ou mais cláusulas `FROM NAMED`, o grafo padrão estará em branco.

Exemplos de **`DESCRIBE`** do SPARQL

Pense nos seguintes dados:

```
PREFIX ex: <https://example.com/>

GRAPH ex:g1 {
    ex:s ex:p1 "a" .
    ex:s ex:p2 "c" .
}
```

```
GRAPH ex:g2 {
    ex:s ex:p3 "b" .
    ex:s ex:p2 "c" .
}

ex:s ex:p3 "d" .
```

Para essa consulta:

```
PREFIX ex: <https://example.com/>
DESCRIBE ?s
FROM ex:g1
FROM NAMED ex:g2
WHERE {
  GRAPH ex:g2 { ?s ?p "b" . }
}
```

O Neptune geraria:

```
ex:s ex:p1 "a" .
ex:s ex:p2 "c" .
```

Aqui, o padrão do grafo `GRAPH ex:g2 { ?s ?p "b" }` é avaliado primeiro, gerando vinculações para `?s` e, depois, a parte `DESCRIBE` é avaliada sobre o grafo padrão, que agora é apenas `ex:g1`.

No entanto, para essa consulta:

```
PREFIX ex: <https://example.com/>
DESCRIBE ?s
FROM NAMED ex:g1
WHERE {
  GRAPH ex:g1 { ?s ?p "a" . }
}
```

O Neptune não geraria nada, porque quando uma cláusula `FROM NAMED` está presente sem nenhuma cláusula `FROM`, o grafo padrão está em branco.

Na seguinte consulta, `DESCRIBE` é usado sem as cláusulas `FROM` ou `FROM NAMED`presentes:

```
PREFIX ex: <https://example.com/>
DESCRIBE ?s
```

```
WHERE {
  GRAPH ex:g1 { ?s ?p "a" . }
}
```

Nessa situação, o grafo padrão é composto por todos os triplos exclusivos na união de todos os grafos nomeados no banco de dados (formalmente, a mesclagem do RDF), então o Neptune geraria:

```
ex:s ex:p1 "a" .
ex:s ex:p2 "c" .
ex:s ex:p3 "b" .
ex:s ex:p3 "d" .
```

API de status de consulta do SPARQL

Para obter o status das consultas SPARQL, use HTTP `GET` ou `POST` para fazer uma solicitação ao endpoint de `https://`*`your-neptune-endpoint`*`:`*`port`*`/sparql/status`.

Parâmetros de solicitação do status de consulta do SPARQL

queryId (opcional)

O ID de uma consulta SPARQL em execução. Exibe apenas o status da consulta especificada.

Sintaxe da resposta do status de consulta do SPARQL

```
{
    "acceptedQueryCount": integer,
    "runningQueryCount": integer,
    "queries": [
      {
        "queryId":"guid",
        "queryEvalStats":
          {
            "subqueries": integer,
            "elapsed": integer,
            "cancelled": boolean
          },
        "queryString": "string"
      }
    ]
}
```

Valores da resposta do status de consulta do SPARQL

aceito QueryCount

O número de consultas aceitas desde o último reinício do mecanismo do Neptune.

correndo QueryCount

O número de consultas SPARQL em execução atualmente.

queries

Uma lista de consultas SPARQL atuais.

queryId

Um ID de GUID da consulta. O Neptune atribui automaticamente esse valor de ID a cada consulta, ou você também pode atribuir seu próprio ID (consulte Injetar um ID personalizado em uma consulta do Gremlin ou do SPARQL no Neptune).

consulta EvalStats

Estatísticas dessa consulta.

subconsultas

Número de subconsultas nesta consulta.

elapsed

O número de milissegundos em que a consulta esteve em execução até agora.

cancelado

Verdadeiro indica que a consulta foi cancelada.

queryString

A consulta enviada.

Exemplo de status de consulta do SPARQL

A seguir está um exemplo de comando de status usando `curl` e HTTP `GET`.

```
curl https://your-neptune-endpoint:port/sparql/status
```

Essa saída mostra uma única consulta em execução.

```
{
    "acceptedQueryCount":9,
    "runningQueryCount":1,
    "queries": [
        {
            "queryId":"fb34cd3e-f37c-4d12-9cf2-03bb741bf54f",
            "queryEvalStats":
                {
                    "subqueries": 0,
                    "elapsed": 29256,
                    "cancelled": false
                },
            "queryString": "SELECT ?s ?p ?o WHERE {?s ?p ?o}"
        }
    ]
}
```

Cancelamento de consulta do SPARQL

Para obter o status das consultas SPARQL, use HTTP `GET` ou `POST` para fazer uma solicitação ao endpoint de `https://your-neptune-endpoint:port/sparql/status`.

Parâmetros de solicitação do cancelamento da consulta do SPARQL

cancelQuery

(Obrigatório) Ordena ao comando de status para cancelar uma consulta. Esse parâmetro não usa um valor.

queryId

(Obrigatório) O ID da consulta SPARQL em execução a ser cancelada.

silent

(Opcional) Se `silent=true`, a consulta em execução será cancelada e o código de resposta HTTP será 200. Se `silent` não estiver presente ou `silent=false`, a consulta será cancelada com um código de status HTTP 500.

Exemplos de cancelamento de consultas do SPARQL

Exemplo 1: cancelamento com **`silent=false`**

Veja a seguir um exemplo do comando de status que usa `curl` para cancelar uma consulta com o parâmetro `silent` definido como `false`:

```
curl https://your-neptune-endpoint:port/sparql/status \
  -d "cancelQuery" \
  -d "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47" \
  -d "silent=false"
```

A menos que a consulta já tenha iniciado os resultados de streaming, a consulta cancelada retornaria um código HTTP 500 com uma resposta como esta:

```
{
  "code": "CancelledByUserException",
  "requestId": "4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47",
  "detailedMessage": "Operation terminated (cancelled by user)"
}
```

Se a consulta já retornou um código HTTP 200 (OK) e iniciou os resultados de streaming antes de ser cancelada, as informações de exceção de tempo limite serão enviadas para o fluxo de saída normal.

Exemplo 2: cancelamento com **`silent=true`**

Veja a seguir um exemplo do mesmo comando de status acima, com a exceção de que o parâmetro `silent` agora está definido como `true`:

```
curl https://your-neptune-endpoint:port/sparql/status \
  -d "cancelQuery" \
  -d "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47" \
  -d "silent=true"
```

Esse comando retornaria a mesma resposta de quando `silent=false`, mas a consulta cancelada agora retornaria um código HTTP 200 com uma resposta como esta:

```
{
  "head" : {
    "vars" : [ "s", "p", "o" ]
```

```
  },
  "results" : {
    "bindings" : [ ]
  }
}
```

Usar o protocolo HTTP do SPARQL 1.1 Graph Store (GSP) no Amazon Neptune

Na recomendação SPARQL 1.1 Graph Store HTTP Protocol, o W3C definiu um protocolo HTTP para gerenciar grafos do RDF. Ele define operações para remover, criar e substituir o conteúdo do grafo do RDF, bem como para adicionar declarações do RDF ao conteúdo existente.

O protocolo de armazenamento de grafos (GSP) oferece uma maneira conveniente de manipular todo o grafo sem precisar escrever consultas do SPARQL complexas.

A partir de Versão: 1.0.5.0 (27/07/2021), o Neptune é totalmente compatível com esse protocolo.

O endpoint do protocolo de armazenamento de grafos (GSP) é:

```
https://your-neptune-cluster:port/sparql/gsp/
```

Para acessar o grafo padrão com o GSP, use:

```
https://your-neptune-cluster:port/sparql/gsp/?default
```

Para acessar um grafo nomeado com o GSP, use:

```
https://your-neptune-cluster:port/sparql/gsp/?graph=named-graph-URI
```

Detalhes especiais da implementação do GSP no Neptune

O Neptune implementa totalmente a recomendação do W3C que define o GSP. No entanto, há algumas situações que a especificação não abrange.

Uma delas é o caso em que uma solicitação `PUT` ou `POST` especifica um ou mais grafos nomeados no corpo da solicitação que diferem do grafo especificado pelo URL da solicitação. Isso só pode acontecer quando o formato RDF do corpo da solicitação é compatível com grafos nomeados, por exemplo, usando `Content-Type: application/n-quads` ou `Content-Type: application/trig`.

Nessa situação, o Neptune adiciona ou atualiza todos os grafos nomeados presentes no corpo, bem como o grafo nomeado especificado no URL.

Por exemplo, suponha que, começando com um banco de dados vazio, você envie uma solicitação PUT para inverter os votos em três grafos. Um, denominado `urn:votes`, contém todos os votos de todos os anos eleitorais. Outros dois, denominados `urn:votes:2005` e `urn:votes:2019`, contêm votos de anos eleitorais específicos. A solicitação e sua carga útil têm a seguinte aparência:

```
PUT "http://your-Neptune-cluster:port/sparql/gsp/?graph=urn:votes"
  Host: example.com
  Content-Type: application/n-quads

  PAYLOAD:

  <urn:JohnDoe> <urn:votedFor> <urn:Labour> <urn:votes:2005>
  <urn:JohnDoe> <urn:votedFor> <urn:Conservative> <urn:votes:2019>
  <urn:JaneSmith> <urn:votedFor> <urn:LiberalDemocrats> <urn:votes:2005>
  <urn:JaneSmith> <urn:votedFor> <urn:Conservative> <urn:votes:2019>
```

Depois que a solicitação é executada, os dados no banco de dados ficam desta forma:

```
<urn:JohnDoe>   <urn:votedFor> <urn:Labour>           <urn:votes:2005>
<urn:JohnDoe>   <urn:votedFor> <urn:Conservative>     <urn:votes:2019>
<urn:JaneSmith> <urn:votedFor> <urn:LiberalDemocrats> <urn:votes:2005>
<urn:JaneSmith> <urn:votedFor> <urn:Conservative>     <urn:votes:2019>
<urn:JohnDoe>   <urn:votedFor> <urn:Labour>           <urn:votes>
<urn:JohnDoe>   <urn:votedFor> <urn:Conservative>     <urn:votes>
<urn:JaneSmith> <urn:votedFor> <urn:LiberalDemocrats> <urn:votes>
<urn:JaneSmith> <urn:votedFor> <urn:Conservative>     <urn:votes>
```

Outra situação ambígua é quando mais de um grafo é especificado no próprio URL da solicitação, usando qualquer um de `PUT`, `POST`, `GET` ou `DELETE`. Por exemplo: .

```
POST "http://your-Neptune-cluster:port/sparql/gsp/?
graph=urn:votes:2005&graph=urn:votes:2019"
```

Ou:

```
GET "http://your-Neptune-cluster:port/sparql/gsp/?default&graph=urn:votes:2019"
```

Nessa situação, o Neptune gera um HTTP 400 com uma mensagem indicando que somente um grafo pode ser especificado no URL da solicitação.

Analisar a execução de consulta do Neptune usando o **`explain`** do SPARQL

O Amazon Neptune adicionou um atributo do SPARQL denominado explain. Esse atributo é uma ferramenta de autoatendimento para compreender a abordagem da execução realizada pelo mecanismo do Neptune. Você o invoca adicionando um parâmetro `explain` a uma chamada HTTP que envia uma consulta do SPARQL.

O recurso `explain` fornece informações sobre a estrutura lógica dos planos de execução da consulta. Você pode usar essas informações para identificar possíveis gargalos de execução e de avaliação. Você pode usar as dicas de consulta para melhorar os planos de execução de suas consultas.

Tópicos

Como o mecanismo de consulta do SPARQL funciona no Neptune

Para usar as informações que o atributo `explain` do SPARQL fornece, você precisa compreender alguns detalhes sobre como o mecanismo de consulta do SPARQL funciona no Amazon Neptune.

O mecanismo converte cada consulta do SPARQL em um pipeline de operadores. A partir do primeiro operador, soluções intermediárias conhecidas como listas de associações fluem por meio desse pipeline de operadores. Você pode considerar uma lista de associações como uma tabela na qual os cabeçalhos da tabela são um subconjunto das variáveis usadas na consulta. Cada linha da tabela representa um resultado, até o ponto de avaliação.

Vamos supor que dois prefixos de namespace foram definidos para nossos dados:

```
  @prefix ex:   <http://example.com> .
```

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
```

Veja a seguir um exemplo de uma lista de associações simples nesse contexto:

```
?person       | ?firstName
-------------------------------------------------------
ex:JaneDoe    | "Jane"
ex:JohnDoe    | "John"
ex:RichardRoe | "Richard"
```

Para cada uma das três pessoas, a lista vincula a variável `?person` a um identificador da pessoa, e a variável `?firstName` ao nome da pessoa.

Em geral, as variáveis poderão permanecer desvinculadas, se, por exemplo, houver uma seleção `OPTIONAL` de uma variável em uma consulta para a qual nenhum valor está presente nos dados.

O operador `PipelineJoin` é um exemplo de mecanismo de consulta do Neptune presente na saída de `explain`. Ele utiliza um conjunto de associações de entrada do operador anterior e junta isso a um padrão triplo, digamos `(?person, foaf:lastName, ?lastName)`. Esta operação usa as associações da variável `?person` em seu fluxo de entrada, substitui-as no padrão triplo e procura triplos no banco de dados.

Quando executado no contexto de associações de entrada da tabela anterior, o `PipelineJoin` deve avaliar três pesquisas, ou seja, o seguinte:

```
(ex:JaneDoe,    foaf:lastName, ?lastName)
(ex:JohnDoe,    foaf:lastName, ?lastName)
(ex:RichardRoe, foaf:lastName, ?lastName)
```

Essa abordagem é chamada de avaliação as-bound. As soluções desse processo de avaliação são juntadas de volta com as soluções de entrada, preenchendo o `?lastName` detectado nas soluções de entrada. Supondo-se que você encontre um sobrenome para todas as três pessoas, o operador deve produzir uma lista de associações de saída que será semelhante à seguinte:

```
?person       | ?firstName | ?lastName
---------------------------------------
ex:JaneDoe    | "Jane"     | "Doe"
ex:JohnDoe    | "John"     | "Doe"
ex:RichardRoe | "Richard"  | "Roe"
```

Essa lista de associações de saída serve, então, como entrada para o próximo operador no pipeline. No final, a saída do último operador no pipeline define o resultado da consulta.

Com frequência, os pipelines de operadores são lineares, no sentido de que cada operador emite soluções para um único operador conectado. No entanto, em alguns casos, eles podem ter estruturas mais complexas. Por exemplo, um operador `UNION` em uma consulta do SPARQL é mapeado para uma operação `Copy`. Essa operação duplica as associações e encaminha as cópias para dois subplanos, uma para o lado esquerdo e a outra para o lado direito da `UNION`.

Para obter mais informações sobre operadores, consulte Operadores SPARQL `explain` no Neptune.

Como usar o **`explain`** do SPARQL para analisar a execução de consultas do Neptune

O atributo `explain` do SPARQL é uma ferramenta de autoatendimento no Amazon Neptune que ajuda a compreender a abordagem da execução realizada pelo mecanismo do Neptune. Para invocar o `explain`, você passa um parâmetro para uma solicitação HTTP ou HTTPS no formulário `explain=`*`mode`*.

O valor do modo pode ser `static`, `dynamic` ou `details`:

- No modo estático, o `explain` imprime somente a estrutura estática do plano de consulta.
- No modo dynamic, o `explain` também inclui aspectos dinâmicos do plano de consulta. Esses aspectos podem incluir o número de associações intermediárias que fluem por meio dos operadores, a proporção de associações de entrada para associações de saída e o tempo total utilizado pelos operadores.
- No modo de detalhes, `explain` imprime as informações mostradas no modo `dynamic` mais detalhes adicionais, como a string de consulta SPARQL real e a contagem de intervalo estimada para o padrão subjacente a um operador de junção.

O Neptune é compatível com o uso do `explain` com todos os três protocolos de acesso de consulta do SPARQL listados na especificação W3C SPARQL 1.1 Protocol, ou seja:

1. HTTP GET
2. HTTP POST usando parâmetros codificados por URL
3. HTTP POST usando parâmetros de texto

Para obter informações sobre o mecanismo de consulta do SPARQL, consulte Como o mecanismo de consulta do SPARQL funciona no Neptune.

Para obter informações sobre o tipo de saída produzido ao invocar o `explain` do SPARQL, consulte Exemplos de invocação do SPARQL `explain` no Neptune.

Exemplos de invocação do SPARQL **`explain`** no Neptune

Os exemplos desta seção mostram os vários tipos de saída que você pode produzir ao invocar o atributo `explain` do SPARQL para analisar a execução de consultas no Amazon Neptune.

Tópicos

Compreensão da saída do explain

Neste exemplo, Jane Doe conhece duas pessoas, ou seja, John Doe e Richard Roe:

```
@prefix ex: <http://example.com> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

ex:JaneDoe foaf:knows ex:JohnDoe .
ex:JohnDoe foaf:firstName "John" .
ex:JohnDoe foaf:lastName "Doe" .
ex:JaneDoe foaf:knows ex:RichardRoe .
ex:RichardRoe foaf:firstName "Richard" .
ex:RichardRoe foaf:lastName "Roe" .
.
```

Para determinar os primeiros nomes de todas as pessoas que Jane Doe conhece, você pode escrever a seguinte consulta:

```
curl http(s)://your_server:your_port/sparql \
```

```
   -d "query=PREFIX foaf: <https://xmlns.com/foaf/0.1/> PREFIX ex: <https://
www.example.com/> \
       SELECT ?firstName WHERE { ex:JaneDoe foaf:knows ?person . ?person
 foaf:firstName ?firstName }" \
   -H "Accept: text/csv"
```

Esta consulta simples retorna o seguinte resultado:

```
firstName
John
Richard
```

Depois, altere o comando `curl` para invocar o `explain` adicionando `-d "explain=dynamic"` e usando o tipo de saída padrão em vez de `text/csv`:

```
 curl http(s)://your_server:your_port/sparql \
   -d "query=PREFIX foaf: <https://xmlns.com/foaf/0.1/> PREFIX ex: <https://
www.example.com/> \
       SELECT ?firstName WHERE { ex:JaneDoe foaf:knows ?person . ?person
 foaf:firstName ?firstName }" \
   -d "explain=dynamic"
```

A consulta agora retorna a saída em formato ASCII formatada (tipo de conteúdo HTTP `text/plain`), que é o tipo de saída padrão:

```
########################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments
            # Mode     # Units In # Units Out # Ratio # Time (ms) #
########################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]
            # -        # 0        # 1         # 0.00  # 0         #
########################################################################################
# 1  # 2      # -      # PipelineJoin      # pattern=distinct(ex:JaneDoe, foaf:knows, ?
person)     # -        # 1        # 2         # 2.00  # 1         #
#    #        #        #                   # joinType=join
            #          #          #           #       #           #
#    #        #        #                   # joinProjectionVars=[?person]
            #          #          #           #       #           #
########################################################################################
# 2  # 3      # -      # PipelineJoin      # pattern=distinct(?person,
 foaf:firstName, ?firstName) # -        # 2        # 2         # 1.00  # 1         #
```

```
#    #        #        #                  # joinType=join
            #          #          #           #       #           #
#    #        #        #                  # joinProjectionVars=[?person, ?firstName]
            #          #          #           #       #           #
#########################################################################################
# 3  # 4      # -      # Projection        # vars=[?firstName]
            # retain   # 2        # 2         # 1.00  # 0         #
#########################################################################################
# 4  # -      # -      # TermResolution    # vars=[?firstName]
            # id2value # 2        # 2         # 1.00  # 1         #
#########################################################################################
```

Para obter detalhes sobre as operações na coluna `Name` e seus argumentos, consulte Operadores explain.

A tabela a seguir descreve a saída linha por linha:

1. A primeira etapa na consulta principal sempre usa o operador `SolutionInjection` para injetar uma solução. Depois, a solução é expandida para o resultado final por meio do processo de avaliação.

   Nesse caso, ele injeta a chamada solução universal `{ }`. Na presença de cláusulas `VALUES` ou de um `BIND`, esta etapa também pode injetar associações mais complexas com as quais começar.

   A coluna `Units Out` indica que o fluxo de saída dessa solução é feito fora do operador. A coluna `Out #1` especifica o operador no qual esse operador alimenta o resultado. Neste exemplo, todos os operadores são conectados ao operador seguinte na tabela.
2. A segunda etapa é um `PipelineJoin`. Ele recebe como entrada a única solução universal (totalmente irrestrita) produzida pelo operador anterior (`Units In := 1`). Ele a une com o padrão de tupla definido por seu argumento `pattern`. Isso corresponde a uma pesquisa simples para o padrão. Nesse caso, o padrão do triplo é definido como o seguinte:

   ```
   distinct( ex:JaneDoe, foaf:knows, ?person )
   ```

   O argumento `joinType := join` indica que esta é uma junção normal (outros tipos incluem junções `optional`, junções `existence check` e assim por diante).

   O argumento `distinct := true` diz que você extrai apenas correspondências distintas do banco de dados (sem duplicatas), e você associa as correspondências distintas à variável `joinProjectionVars := ?person`, desduplicada.

O fato de que o valor da coluna `Units Out` é 2 indica que há duas soluções no fluxo de saída. Especificamente, essas são as associações da variável `?person`, que refletem as duas pessoas que os dados mostram que Jane Doe conhece:

```
?person
-------------
ex:JohnDoe
ex:RichardRoe
```

3. As duas soluções da etapa 2 fluem como entrada (`Units In := 2`) no segundo `PipelineJoin`. Esse operador une as duas soluções anteriores com o seguinte padrão de triplo:

   ```
   distinct(?person, foaf:firstName, ?firstName)
   ```

   Sabe-se que a variável `?person` está associada a `ex:JohnDoe` ou a `ex:RichardRoe` pela solução de entrada do operador. Considerando isso, o `PipelineJoin` extrai os nomes, John e Richard. As duas soluções de saída (Units Out: = 2) são as seguintes:

   ```
   ?person       | ?firstName
   ---------------------------
   ex:JohnDoe    | John
   ex:RichardRoe | Richard
   ```

4. O próximo operador da projeção usa como entrada as duas soluções da etapa 3 (`Units In := 2`) e projeta para a variável `?firstName`. Isso elimina todas as outras associações da variável nos mapeamentos e passa as duas associações (`Units Out := 2`):

   ```
   ?firstName
   ----------
   John
   Richard
   ```

5. Para melhorar o desempenho, o Neptune opera sempre que possível em identificadores internos que ele atribui aos termos como URIs e literais de strings, e não nas próprias strings. O operador final, `TermResolution`, executa um mapeamento desses identificadores internos de volta para as strings dos termos correspondentes.

   Na avaliação de consultas normais (sem explain), o resultado calculado pelo último operador é, então, serializado para o formato de serialização solicitado e transmitido para o cliente.

Exemplo de saída do modo de detalhes

> **Note**
>
> O modo de detalhes explain do SPARQL está disponível a partir da versão 1.0.2.1 do mecanismo do Neptune.

Vamos supor que você execute a mesma consulta que a anterior no modo de detalhes, em vez de no modo dinâmico:

```
 curl http(s)://your_server:your_port/sparql \
   -d "query=PREFIX foaf: <https://xmlns.com/foaf/0.1/> PREFIX ex: <https://
www.example.com/> \
       SELECT ?firstName WHERE { ex:JaneDoe foaf:knows ?person . ?person
 foaf:firstName ?firstName }" \
   -d "explain=details"
```

Como mostra este exemplo, a saída é a mesma com alguns detalhes adicionais, como a string de consulta na parte superior da saída e a contagem `patternEstimate` para o operador `PipelineJoin`:

```
Query:
PREFIX foaf: <https://xmlns.com/foaf/0.1/> PREFIX ex: <https://www.example.com/>
SELECT ?firstName WHERE { ex:JaneDoe foaf:knows ?person . ?person foaf:firstName ?
firstName }

#############################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments
            # Mode     # Units In # Units Out # Ratio # Time (ms) #
#############################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]
            # -        # 0         # 1          # 0.00  # 0         #
#############################################################################################
# 1  # 2      # -      # PipelineJoin      # pattern=distinct(ex:JaneDoe, foaf:knows, ?
person)     # -        # 1         # 2          # 2.00  # 13        #
#    #        #        #                   # joinType=join
            #          #          #           #       #           #
#    #        #        #                   # joinProjectionVars=[?person]
            #          #          #           #       #           #
#    #        #        #                   # patternEstimate=2
            #          #          #           #       #           #
```

```
#################################################################################################
# 2  # 3      # -      # PipelineJoin      # pattern=distinct(?person,
 foaf:firstName, ?firstName) # -        # 2        # 2         # 1.00  # 3         #
#    #        #        #                   # joinType=join
            #          #          #           #       #           #
#    #        #        #                   # joinProjectionVars=[?person, ?firstName]
            #          #          #           #       #           #
#    #        #        #                   # patternEstimate=2
            #          #          #           #       #           #
#################################################################################################
# 3  # 4      # -      # Projection        # vars=[?firstName]
            # retain   # 2        # 2         # 1.00  # 1         #
#################################################################################################
# 4  # -      # -      # TermResolution    # vars=[?firstName]
            # id2value # 2        # 2         # 1.00  # 7         #
#################################################################################################
```

Exemplo de saída de modo estático

Vamos supor que você execute a mesma consulta que a anterior no modo estático (o padrão), em vez de no modo de detalhes:

```
 curl http(s)://your_server:your_port/sparql \
   -d "query=PREFIX foaf: <https://xmlns.com/foaf/0.1/> PREFIX ex: <https://
www.example.com/> \
       SELECT ?firstName WHERE { ex:JaneDoe foaf:knows ?person . ?person
 foaf:firstName ?firstName }" \
   -d "explain=static"
```

Como mostra o exemplo, a saída é a mesma, com a exceção de que ela omite as últimas três colunas:

```
#################################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments
            # Mode     #
#################################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]
            # -        #
#################################################################################################
# 1  # 2      # -      # PipelineJoin      # pattern=distinct(ex:JaneDoe, foaf:knows, ?
person)     # -        #
#    #        #        #                   # joinType=join
            #          #
```

```
#    #        #        #                  # joinProjectionVars=[?person]
            #          #
##############################################################################################
# 2  # 3      # -      # PipelineJoin      # pattern=distinct(?person,
 foaf:firstName, ?firstName) # -        #
#    #        #        #                  # joinType=join
            #          #
#    #        #        #                  # joinProjectionVars=[?person, ?firstName]
            #          #
##############################################################################################
# 3  # 4      # -      # Projection        # vars=[?firstName]
            # retain   #
##############################################################################################
# 4  # -      # -      # TermResolution    # vars=[?firstName]
            # id2value #
##############################################################################################
```

Diferentes formas de codificar parâmetros

As consultas de exemplo a seguir ilustram duas formas diferentes de codificar parâmetros ao invocar o `explain` do SPARQL.

Usar a codificação de URL: este exemplo usa a codificação de URL de parâmetros e especifica a saída dinâmica:

```
curl -XGET "http(s)://your_server:your_port/sparql?query=SELECT%20*%20WHERE%20%7B
%20%3Fs%20%3Fp%20%3Fo%20%7D%20LIMIT%20%31&explain=dynamic"
```

Especificar os parâmetros diretamente: esta consulta é igual à anterior, com a exceção de que ela transmite os parâmetros diretamente por meio de POST:

```
curl http(s)://your_server:your_port/sparql \
  -d "query=SELECT * WHERE { ?s ?p ?o } LIMIT 1" \
  -d "explain=dynamic"
```

Outros tipos de saída além de texto/sem formatação

Os exemplos anteriores usam o tipo de saída de `text/plain` padrão. O Neptune também pode formatar a saída de `explain` SPARQL em dois outros formatos do tipo MIME, ou seja, `text/csv` e `text/html`. Você as invoca configurando o cabeçalho `Accept` de HTTP, o que pode ser feito usando o sinalizador `-H` no `curl`, da seguinte forma:

```
  -H "Accept: output type"
```

Veja alguns exemplos:

Saída **text/csv**

Essa consulta chama a saída CSV de tipo MIME especificando `-H "Accept: text/csv"`:

```
curl http(s)://your_server:your_port/sparql \
  -d "query=SELECT * WHERE { ?s ?p ?o } LIMIT 1" \
  -d "explain=dynamic" \
  -H "Accept: text/csv"
```

O formato CSV, que é útil para importação em uma planilha ou banco de dados, separa os campos em cada linha do `explain` com ponto e vírgulas (`;`), da seguinte forma:

```
ID;Out #1;Out #2;Name;Arguments;Mode;Units In;Units Out;Ratio;Time (ms)
0;1;-;SolutionInjection;solutions=[{}];-;0;1;0.00;0
1;2;-;PipelineJoin;pattern=distinct(?s, ?p, ?o),joinType=join,joinProjectionVars=[?s, ?
p, ?o];-;1;6;6.00;1
2;3;-;Projection;vars=[?s, ?p, ?o];retain;6;6;1.00;2
3;-;-;Slice;limit=1;-;1;1;1.00;1
```

Saída **text/html**

Se você especificar `-H "Accept: text/html"`, o `explain` gerará uma tabela HTML:

```
<!DOCTYPE html>
<html>
  <body>
    <table border="1px">
      <thead>
        <tr>
          <th>ID</th>
          <th>Out #1</th>
          <th>Out #2</th>
          <th>Name</th>
          <th>Arguments</th>
          <th>Mode</th>
          <th>Units In</th>
```

```
          <th>Units Out</th>
          <th>Ratio</th>
          <th>Time (ms)</th>
        </tr>
      </thead>

      <tbody>
        <tr>
          <td>0</td>
          <td>1</td>
          <td>-</td>
          <td>SolutionInjection</td>
          <td>solutions=[{}]</td>
          <td>-</td>
          <td>0</td>
          <td>1</td>
          <td>0.00</td>
          <td>0</td>
        </tr>

        <tr>
          <td>1</td>
          <td>2</td>
          <td>-</td>
          <td>PipelineJoin</td>
          <td>pattern=distinct(?s, ?p, ?o)<br>
              joinType=join<br>
              joinProjectionVars=[?s, ?p, ?o]</td>
          <td>-</td>
          <td>1</td>
          <td>6</td>
          <td>6.00</td>
          <td>1</td>
        </tr>

        <tr>
          <td>2</td>
          <td>3</td>
          <td>-</td>
          <td>Projection</td>
          <td>vars=[?s, ?p, ?o]</td>
          <td>retain</td>
          <td>6</td>
          <td>6</td>
```

```
          <td>1.00</td>
          <td>2</td>
        </tr>

        <tr>
          <td>3</td>
          <td>-</td>
          <td>-</td>
          <td>Slice</td>
          <td>limit=1</td>
          <td>-</td>
          <td>1</td>
          <td>1</td>
          <td>1.00</td>
          <td>1</td>
        </tr>
      </tbody>
    </table>
  </body>
</html>
```

O HTML renderizará em um navegador algo como o seguinte:

ID	Out #1	Out #2	Name	Arguments	Mode	Units In	Units Out	Ratio	Time (ms)
0	1	-	SolutionInjection	solutions=[{}]	-	0	1	0.00	0
1	2	-	PipelineJoin	pattern=distinct(?s, ?p, ?o) joinType=join joinProjectionVars=[?s, ?p, ?o]	-	1	6	6.00	1
2	3	-	Projection	vars=[?s, ?p, ?o]	retain	6	6	1.00	2
3	-	-	Slice	limit=1	-	1	1	1.00	1

Exemplo de saída de **`explain`** SPARQL quando o DFE está habilitado

Veja um exemplo de saída de `explain` SPARQL quando o mecanismo de consulta alternativo do DFE no Neptune está habilitado:

```
################################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments


 # Mode     # Units In # Units Out # Ratio # Time (ms) #
################################################################################################
```

```
# 0  # 1      # -      # SolutionInjection # solutions=[{}]


 # -        # 0        # 1         # 0.00  # 0         #
##########################################################################################
# 1  # 2      # -      # HashIndexBuild    # solutionSet=solutionSet1


 # -        # 1        # 1         # 1.00  # 22        #
#    #        #        #                   # joinVars=[]


 #          #          #           #       #           #
#    #        #        #                   # sourceType=pipeline


 #          #          #           #       #           #
##########################################################################################
# 2  # 3      # -      # DFENode            # DFE Stats=


      # -        # 101      # 100       # 0.99  # 32        #
#    #        #        #                   # ====> DFE execution time (measured by
 DFEQueryEngine)

     #          #          #           #       #           #
#    #        #        #                   # accepted [micros]=127


 #          #          #           #       #           #
#    #        #        #                   # ready [micros]=2


 #          #          #           #       #           #
#    #        #        #                   # running [micros]=5627


 #          #          #           #       #           #
#    #        #        #                   # finished [micros]=0


 #          #          #           #       #           #
#    #        #        #                   #
```

```
 #          #          #           #       #           #
#    #        #        #                   #


 #          #          #           #       #           #
#    #        #        #                   # ===> DFE execution time (measured in
 DFENode)

      #          #          #           #       #           #
#    #        #        #                   # -> setupTime [ms]=1


 #          #          #           #       #           #
#    #        #        #                   # -> executionTime [ms]=14


 #          #          #           #       #           #
#    #        #        #                   # -> resultReadTime [ms]=0


 #          #          #           #       #           #
#    #        #        #                   #


 #          #          #           #       #           #
#    #        #        #                   #


 #          #          #           #       #           #
#    #        #        #                   # ===> Static analysis statistics


 #          #          #           #       #           #
#    #        #        #                   # --> 35907 micros spent in parser.


 #          #          #           #       #           #
#    #        #        #                   # --> 7643 micros spent in range count
 estimation

      #          #          #           #       #           #
#    #        #        #                   # --> 2895 micros spent in value resolution
```

```
 #          #          #           #       #           #
#    #        #        #                   #


 #          #          #           #       #           #
#    #        #        #                   # --> 39974925 micros spent in optimizer
 loop

    #          #          #           #       #           #
#    #        #        #                   #


 #          #          #           #       #           #
#    #        #        #                   #


 #          #          #           #       #           #
#    #        #        #                   # DFEJoinGroupNode[ children={


  #          #          #           #       #           #
#    #        #        #                   #   DFEPatternNode[(?1, TERM[117442062], ?
2, ?3) . project DISTINCT[?1, ?2] {rangeCountEstimate=100},

  #          #          #           #       #           #
#    #        #        #                   #     OperatorInfoWithAlternative[


  #          #          #           #       #           #
#    #        #        #                   #       rec=OperatorInfo[


  #          #          #           #       #           #
#    #        #        #                   #         type=INCREMENTAL_PIPELINE_JOIN,


 #          #          #           #       #           #
#    #        #        #                   #
 costEstimates=OperatorCostEstimates[

                                  #          #          #           #       #
  #
```

```
#    #        #        #                   #
 costEstimate=OperatorCostEstimate[in=1.0000,out=100.0000,io=0.0002,comp=0.0000,mem=0],

                                  #          #          #           #       #
 #
#    #        #        #                   #
 worstCaseCostEstimate=OperatorCostEstimate[in=1.0000,out=100.0000,io=0.0002,comp=0.0000,mem=0]

                       #          #          #           #       #           #
#    #        #        #                   #       alt=OperatorInfo[


  #          #          #           #       #           #
#    #        #        #                   #         type=INCREMENTAL_HASH_JOIN,


 #          #          #           #       #           #
#    #        #        #                   #
 costEstimates=OperatorCostEstimates[

                                  #          #          #           #       #
  #
#    #        #        #                   #
 costEstimate=OperatorCostEstimate[in=1.0000,out=100.0000,io=0.0003,comp=0.0000,mem=3212],

                                #          #          #           #       #           #
#    #        #        #                   #
 worstCaseCostEstimate=OperatorCostEstimate[in=1.0000,out=100.0000,io=0.0003,comp=0.0000,mem=32

                  #          #          #           #       #           #
#    #        #        #                   #   DFEPatternNode[(?1, TERM[150997262], ?
4, ?5) . project DISTINCT[?1, ?4] {rangeCountEstimate=100},

  #          #          #           #       #           #
#    #        #        #                   #     OperatorInfoWithAlternative[


  #          #          #           #       #           #
#    #        #        #                   #       rec=OperatorInfo[


  #          #          #           #       #           #
#    #        #        #                   #         type=INCREMENTAL_HASH_JOIN,
```

```
 #          #          #           #       #           #
#    #        #        #                   #
 costEstimates=OperatorCostEstimates[

                                    #          #          #           #       #
  #
#    #        #        #                   #
 costEstimate=OperatorCostEstimate[in=100.0000,out=100.0000,io=0.0003,comp=0.0000,mem=6400],

                              #          #          #           #       #           #
#    #        #        #                   #
 worstCaseCostEstimate=OperatorCostEstimate[in=100.0000,out=100.0000,io=0.0003,comp=0.0000,mem=

                 #          #          #           #       #           #
#    #        #        #                   #       alt=OperatorInfo[


  #          #          #           #       #           #
#    #        #        #                   #         type=INCREMENTAL_PIPELINE_JOIN,


 #          #          #           #       #           #
#    #        #        #                   #
 costEstimates=OperatorCostEstimates[

                                    #          #          #           #       #
  #
#    #        #        #                   #
 costEstimate=OperatorCostEstimate[in=100.0000,out=100.0000,io=0.0010,comp=0.0000,mem=0],

                                 #          #          #           #       #           #
#    #        #        #                   #
 worstCaseCostEstimate=OperatorCostEstimate[in=100.0000,out=100.0000,io=0.0010,comp=0.0000,mem=

                   #          #          #           #       #           #
#    #        #        #                   # },


 #          #          #           #       #           #
#    #        #        #                   # ]


 #          #          #           #       #           #
```

```
#    #         #         #                  #

 #          #          #          #       #           #
#    #         #         #                  # ===> DFE configuration:

 #          #          #          #       #           #
#    #         #         #                  # solutionChunkSize=5000

 #          #          #          #       #           #
#    #         #         #                  # ouputQueueSize=20

 #          #          #          #       #           #
#    #         #         #                  # numComputeCores=3

 #          #          #          #       #           #
#    #         #         #                  # maxParallelIO=10

 #          #          #          #       #           #
#    #         #         #                  # numInitialPermits=12

 #          #          #          #       #           #
#    #         #         #                  #

 #          #          #          #       #           #
#    #         #         #                  #

 #          #          #          #       #           #
#    #         #         #                  # ====> DFE configuration (reported back)

 #          #          #          #       #           #
#    #         #         #                  # numComputeCores=3

 #          #          #          #       #           #
```

```
#    #        #        #                  # maxParallelIO=2


 #          #          #           #       #           #
#    #        #        #                  # numInitialPermits=12


 #          #          #           #       #           #
#    #        #        #                  #


 #          #          #           #       #           #
#    #        #        #                  # ===> Statistics & operator histogram


 #          #          #           #       #           #
#    #        #        #                  # ==> Statistics


 #          #          #           #       #           #
#    #        #        #                  # -> 3741 / 3668 micros total elapsed (incl.
 wait / excl. wait)

 #          #          #           #       #           #
#    #        #        #                  # -> 3741 / 3 millis total elapse (incl.
 wait / excl. wait)

    #          #          #           #       #           #
#    #        #        #                  # -> 3741 / 0 secs total elapsed (incl.
 wait / excl. wait)

     #          #          #           #       #           #
#    #        #        #                  # ==> Operator histogram


 #          #          #           #       #           #
#    #        #        #                  # -> 47.66% of total time (excl. wait):
 pipelineScan (2 instances)

     #          #          #           #       #           #
#    #        #        #                  # -> 10.99% of total time (excl. wait):
 merge (1 instances)

     #          #          #           #       #           #
```

```
#    #        #        #                  # -> 41.17% of total time (excl. wait):
 symmetricHashJoin (1 instances)

     #          #          #           #       #           #
#    #        #        #                  # -> 0.19% of total time (excl. wait): drain
 (1 instances)

 #          #          #           #       #           #
#    #        #        #                  #


 #          #          #           #       #           #
#    #        #        #                  # nodeId | out0   | out1 | opName
   | args                                          | rowsIn | rowsOut | chunksIn |
 chunksOut | elapsed* | outWait | outBlocked | ratio    | rate* [M/s] | rate [M/s] | %
    #          #          #           #       #           #
#    #        #        #                  # ------ | ------ | ---- | -----------------
 | ------------------------------------------------ | ------ | ------- | -------- |
 --------- | -------- | ------- | ---------- | -------- | ----------- | ---------- |
 ----- #          #          #           #       #           #
#    #        #        #                  # node_0 | node_2 | -    | pipelineScan
  | (?1, TERM[117442062], ?2, ?3) DISTINCT [?1, ?2]  | 0      | 100     | 0        | 1
        | 874      | 0       | 0          | Infinity | 0.1144      | 0.1144     | 23.83
 #          #          #           #       #           #
#    #        #        #                  # node_1 | node_2 | -    | pipelineScan
  | (?1, TERM[150997262], ?4, ?5) DISTINCT [?1, ?4]  | 0      | 100     | 0        | 1
        | 874      | 0       | 0          | Infinity | 0.1144      | 0.1144     | 23.83
 #          #          #           #       #           #
#    #        #        #                  # node_2 | node_4 | -    | symmetricHashJoin
 |                                                 | 200    | 100     | 2        | 2
        | 1510     | 73      | 0          | 0.50     | 0.0662      | 0.0632     | 41.17
 #          #          #           #       #           #
#    #        #        #                  # node_3 | -      | -    | drain
  |                                                 | 100    | 0       | 1        | 0
        | 7        | 0       | 0          | 0.00     | 0.0000      | 0.0000     | 0.19
 #          #          #           #       #           #
#    #        #        #                  # node_4 | node_3 | -    | merge
  |                                                 | 100    | 100     | 2        | 1
        | 403      | 0       | 0          | 1.00     | 0.2481      | 0.2481     | 10.99
 #          #          #           #       #           #
##########################################################################################
# 3  # 4      # -      # HashIndexJoin     # solutionSet=solutionSet1
```

```
 # -        # 100      # 100       # 1.00  # 4         #
#    #        #        #                   # joinType=join


 #          #          #           #       #           #
#########################################################################################
# 4  # 5      # -      # Distinct          # vars=[?s, ?o, ?o1]


 # -        # 100      # 100       # 1.00  # 9         #
#########################################################################################
# 5  # 6      # -      # Projection        # vars=[?s, ?o, ?o1]


 # retain   # 100      # 100       # 1.00  # 2         #
#########################################################################################
# 6  # -      # -      # TermResolution    # vars=[?s, ?o, ?o1]


 # id2value # 100      # 100       # 1.00  # 11        #
#########################################################################################
```

Operadores SPARQL **`explain`** no Neptune

As seções a seguir descrevem os operadores e os parâmetros para o atributo `explain` do SPARQL que está disponível no Amazon Neptune.

> **Important**
>
> O recurso `explain` do SPARQL ainda está sendo refinado. Os operadores e os parâmetros documentados aqui podem ser alterados em versões futuras.

Tópicos

- Operador Aggregation
- Operador ConditionalRouting
- Operador Copy
- Operador DFENode
- Operador Distinct

- Operador Federation
- Operador Filter
- Operador HashIndexBuild
- Operador HashIndexJoin
- Operador MergeJoin
- Operador NamedSubquery
- Operador PipelineJoin
- Operador PipelineCountJoin
- Operador PipelinedHashIndexJoin
- Operador Projection
- Operador PropertyPath
- Operador TermResolution
- Operador Slice
- Operador SolutionInjection
- Operador Sort
- Operador VariableAlignment

Operador **`Aggregation`**

Executa uma ou mais agregações, implementando a semântica dos operadores de agregação do SPARQL, como `count`, `max`, `min`, `sum` e assim por diante.

`Aggregation` é fornecido com agrupamento opcional usando cláusulas `groupBy` e restrições `having` opcionais.

Argumentos

- `groupBy`: (opcional) fornece uma cláusula `groupBy` que especifica a sequência de expressões de acordo com a qual as soluções de entrada são agrupadas.
- `aggregates`: (obrigatório) especifica uma lista ordenada de expressões de agregação.
- `having`: (opcional) adiciona restrições para filtragem em grupos, como implícito pela cláusula `having` na consulta do SPARQL.

Operador **`ConditionalRouting`**

Roteia soluções de entrada com base em uma determinada condição. As soluções que atendem à condição são roteadas para o ID do operador referenciado por `Out #1`, enquanto as soluções que não atendem são roteadas para o operador referenciado por `Out #2`.

Argumentos

- `condition`: (obrigatório) a condição de roteamento.

Operador **`Copy`**

Delega o fluxo da solução conforme especificado pelo modo especificado.

Modos

- `forward`: encaminha as soluções para o operador posterior identificado por `Out #1`.
- `duplicate`: duplica as soluções e as encaminha para cada um dos dois operadores identificados por `Out #1` e `Out #2`.

O `Copy` não tem argumentos.

Operador **`DFENode`**

Esse operador é uma abstração do plano executado pelo mecanismo de consulta alternativa do DFE. O plano detalhado do DFE é descrito nos argumentos para esse operador. No momento, o argumento está sobrecarregado para conter as estatísticas detalhadas de runtime do plano do DFE. Ele contém o tempo gasto nas várias etapas da execução da consulta pelo DFE.

A árvore de sintaxe abstrata (AST) de lógica otimizada para o plano de consulta do DFE é impressa com informações sobre os tipos de operadores que foram considerados durante o planejamento e os melhores e piores custos associados à execução dos operadores. O AST consiste nos seguintes tipos de nós no momento:

- `DFEJoinGroupNode`: representa uma junção de um ou mais `DFEPatternNodes`.
- `DFEPatternNode`: encapsula um padrão subjacente usando as tuplas correspondentes que se projetam para fora do banco de dados subjacente.

A subseção, `Statistics & Operator histogram`, contém detalhes sobre o tempo de execução do plano `DataflowOp` e o detalhamento do tempo de CPU usado por cada operador. Abaixo disso, há uma tabela que imprime estatísticas detalhadas de runtime do plano executado pelo DFE.

> **Note**
>
> O DFE é um atributo experimental lançado no modo de laboratório, portanto, o formato exato da saída de `explain` pode mudar.

Operador **`Distinct`**

Calcula a projeção distinta em um subconjunto das variáveis, eliminando duplicatas. Como resultado, o número de soluções do fluxo de entrada é maior ou igual ao número de soluções do fluxo de saída.

Argumentos

- `vars`: (obrigatório) as variáveis às quais aplicar a projeção `Distinct`.

Operador **`Federation`**

Transmite uma consulta especificada para um endpoint SPARQL remoto especificado.

Argumentos

- `endpoint`: (obrigatório) o URL do endpoint na declaração `SERVICE` do SPARQL. Pode ser uma string constante ou, se o endpoint da consulta for determinado com base em uma variável dentro da mesma consulta, pode ser o nome da variável.
- `query`: (obrigatório) a string de consulta reconstruída a ser enviada ao endpoint remoto. O mecanismo adiciona prefixos padrão a essa consulta, mesmo quando o cliente não especifica nenhum.
- `silent`: (obrigatório) um booliano que indica se a palavra-chave `SILENT` apareceu após a palavra-chave. `SILENT` ordena ao mecanismo para não causar falha na consulta inteira, mesmo que a parte remota de `SERVICE` falhe.

Operador **`Filter`**

Filtra as soluções de entrada. Somente as soluções que atendem à condição do filtro são encaminhadas para o operador upstream, e todas as outras são descartadas.

Argumentos

- `condition`: (obrigatório) a condição de filtro.

Operador **`HashIndexBuild`**

Usa uma lista de associações e as transfere para um índice de hash cujo nome é definido pelo argumento `solutionSet`. Normalmente, os operadores subsequentes executam junções nesse conjunto de soluções, referindo-se a ele por esse nome.

Argumentos

- `solutionSet`: (obrigatório) o nome do conjunto de soluções do índice de hash.
- `sourceType`: (obrigatório) o tipo da origem a partir da qual as associações a serem armazenadas no índice de hash são obtidas:
  - `pipeline`: transfere as soluções de entrada do operador posterior para o pipeline de operadores no índice de hash.
  - `binding set`: transfere o conjunto de associações fixas especificado pelo argumento `sourceBindingSet` ao índice de hash.
- `sourceBindingSet`: (opcional) se o valor do argumento `sourceType` for `binding set`, esse argumento especificará o conjunto de associações estáticas a serem transferidas ao índice de hash.

Operador **`HashIndexJoin`**

Une as soluções de entrada no conjunto de soluções do índice de hash identificado pelo argumento `solutionSet`.

Argumentos

- `solutionSet`: (obrigatório) o nome do conjunto de soluções no qual fazer a junção.
  Esse deve ser um índice de hash que foi criado em uma etapa anterior usando o operador `HashIndexBuild`.
- `joinType`: (obrigatório) o tipo de junção a ser executada:
  - `join`: uma junção normal, que exige uma correspondência exata entre todas as variáveis compartilhadas.
  - `optional`: uma junção `optional` que usa a semântica do operador `OPTIONAL` do SPARQL.

  - `minus`: uma operação `minus` mantém um mapeamento para o qual não existe nenhum parceiro de junção usando o operador `MINUS` do SPARQL.
  - `existence check`: confere se há um parceiro de junção e associa a variável `existenceCheckResultVar` ao resultado dessa verificação.
- `constraints`: (opcional) restrições adicionais de junção que são consideradas durante a junção. Junções que não atendem a essas restrições são descartadas.
- `existenceCheckResultVar`: (opcional) Usado apenas para junções em que `joinType` é igual a `existence check` (veja o argumento `joinType` anterior).

Operador **`MergeJoin`**

Uma junção de mesclagem em vários conjuntos de soluções, conforme identificado pelo argumento `solutionSets`.

Argumentos

- `solutionSets`: (obrigatório) os conjuntos de soluções a serem unidos.

Operador **`NamedSubquery`**

Aciona a avaliação da subconsulta identificada pelo argumento `subQuery` e transfere o resultado para o conjunto de soluções especificado pelo argumento `solutionSet`. As soluções de entrada para o operador são encaminhadas para a subconsulta e então para o próximo operador.

Argumentos

- `subQuery`: (obrigatório) o nome da subconsulta a ser avaliada. A subconsulta é renderizada explicitamente na saída.
- `solutionSet`: (obrigatório) o nome do conjunto de soluções no qual armazenar o resultado da subconsulta.

Operador **`PipelineJoin`**

Recebe como entrada a saída do operador anterior e junta-a no padrão de tupla definido pelo argumento `pattern`.

Argumentos

- `pattern`— (Obrigatório) O padrão, que assume a forma de uma tupla e subject-predicate-object, opcionalmente, de gráfico que está por trás da junção. Se `distinct` for especificado para o padrão, a junção extrairá apenas as soluções distintas das variáveis de projeção especificadas pelo argumento `projectionVars`, em vez de todas as soluções correspondentes.
- `inlineFilters`: (opcional) um conjunto de filtros a serem aplicados às variáveis no padrão. O padrão é avaliado em conjunto com esses filtros.
- `joinType`: (obrigatório) o tipo de junção a ser executada:
  - `join`: uma junção normal, que exige uma correspondência exata entre todas as variáveis compartilhadas.
  - `optional`: uma junção `optional` que usa a semântica do operador `OPTIONAL` do SPARQL.
  - `minus`: uma operação `minus` mantém um mapeamento para o qual não existe nenhum parceiro de junção usando o operador `MINUS` do SPARQL.
  - `existence check`: confere se há um parceiro de junção e associa a variável `existenceCheckResultVar` ao resultado dessa verificação.
- `constraints`: (opcional) restrições adicionais de junção que são consideradas durante a junção. Junções que não atendem a essas restrições são descartadas.
- `projectionVars`: (opcional) as variáveis da projeção. Usado em combinação com `distinct := true` para impor a extração de projeções distintas em um conjunto de variáveis especificado.
- `cutoffLimit`: (opcional) um limite de corte para o número de parceiros de junção extraídos. Embora não exista nenhum limite por padrão, você pode definir isso como 1 ao executar junções para implementar cláusulas `FILTER (NOT) EXISTS`, em que é suficiente provar ou refutar que existe um parceiro de junção.

Operador **`PipelineCountJoin`**

Variante do `PipelineJoin`. Em vez de juntar, ele simplesmente conta os parceiros da junção correspondente e associa a contagem à variável especificada pelo argumento `countVar`.

Argumentos

- `countVar`: (obrigatório) a variável à qual o resultado da contagem, ou seja, o número de parceiros da junção, deve ser associado.

- `pattern`— (Obrigatório) O padrão, que assume a forma de uma tupla e subject-predicate-object, opcionalmente, de gráfico que está por trás da junção. Se `distinct` for especificado para o padrão, a junção extrairá apenas as soluções distintas das variáveis de projeção especificadas pelo argumento `projectionVars`, em vez de todas as soluções correspondentes.
- `inlineFilters`: (opcional) um conjunto de filtros a serem aplicados às variáveis no padrão. O padrão é avaliado em conjunto com esses filtros.
- `joinType`: (obrigatório) o tipo de junção a ser executada:
  - `join`: uma junção normal, que exige uma correspondência exata entre todas as variáveis compartilhadas.
  - `optional`: uma junção `optional` que usa a semântica do operador `OPTIONAL` do SPARQL.
  - `minus`: uma operação `minus` mantém um mapeamento para o qual não existe nenhum parceiro de junção usando o operador `MINUS` do SPARQL.
  - `existence check`: confere se há um parceiro de junção e associa a variável `existenceCheckResultVar` ao resultado dessa verificação.
- `constraints`: (opcional) restrições adicionais de junção que são consideradas durante a junção. Junções que não atendem a essas restrições são descartadas.
- `projectionVars`: (opcional) as variáveis da projeção. Usado em combinação com `distinct := true` para impor a extração de projeções distintas em um conjunto de variáveis especificado.
- `cutoffLimit`: (opcional) um limite de corte para o número de parceiros de junção extraídos. Embora não exista nenhum limite por padrão, você pode definir isso como 1 ao executar junções para implementar cláusulas `FILTER (NOT) EXISTS`, em que é suficiente provar ou refutar que existe um parceiro de junção.

Operador **`PipelinedHashIndexJoin`**

Este é um índice de hash de all-in-one construção e um operador de junção. Ele pega uma lista de associações, as agrupa em um índice de hash e, depois, une as soluções de entrada com o índice de hash.

Argumentos

- `sourceType`: (obrigatório) o tipo da origem a partir da qual as associações a serem armazenadas no índice de hash são obtidas. Um de:

  - `pipeline`: faz com que o `PipelinedHashIndexJoin` transfira as soluções de entrada do operador posterior no pipeline de operadores para o índice de hash.
  - `binding set`: faz com que `PipelinedHashIndexJoin` transfira o conjunto de associações fixas especificado pelo argumento `sourceBindingSet` para o índice de hash.
- `sourceSubQuery` : (opcional) se o valor do argumento `sourceType` for `pipeline`, esse argumento especificará a subconsulta avaliada e agrupada no índice de hash.
- `sourceBindingSet` : (opcional) se o valor do argumento `sourceType` for `binding set`, esse argumento especificará o conjunto de associações estáticas a serem transferidas para o índice de hash.
- `joinType`: (obrigatório) o tipo de junção a ser executada:
  - `join`: uma junção normal, que exige uma correspondência exata entre todas as variáveis compartilhadas.
  - `optional`: uma junção `optional` que usa a semântica do operador `OPTIONAL` do SPARQL.
  - `minus`: uma operação `minus` mantém um mapeamento para o qual não existe nenhum parceiro de junção usando o operador `MINUS` do SPARQL.
  - `existence check`: confere se há um parceiro de junção e associa a variável `existenceCheckResultVar` ao resultado dessa verificação.
- `existenceCheckResultVar`: (opcional) usado apenas para junções em que `joinType` é igual a `existence check` (veja o argumento joinType acima).

Operador **`Projection`**

Projeta sobre um subconjunto de variáveis. O número de soluções do fluxo de entrada é igual ao número de soluções do fluxo de saída, mas a forma da solução é diferente, dependendo da configuração do modo.

Modos

- `retain`: reter nas soluções apenas as variáveis que são especificadas pelo argumento `vars`.
- `drop`: descartar todas as variáveis especificadas pelo argumento `vars`.

Argumentos

- `vars`: (obrigatório) as variáveis a serem retidas ou descartadas, dependendo da configuração do modo.

Operador **`PropertyPath`**

Habilita caminhos de propriedades recursivos, como + ou *. O Neptune implementa uma abordagem de iteração de ponto fixo com base em um modelo especificado pelo argumento `iterationTemplate`. As variáveis conhecidas do lado esquerdo ou direito são associadas no modelo para cada iteração de ponto fixo, até que nenhuma nova solução possa ser encontrada.

Argumentos

- `iterationTemplate`: (obrigatório) o nome do modelo de subconsulta usado para implementar a iteração de ponto fixo.
- `leftTerm`: (obrigatório) o termo (variável ou constante) no lado esquerdo do caminho da propriedade.
- `rightTerm`: (obrigatório) o termo (variável ou constante) no lado direito do caminho da propriedade.
- `lowerBound`: (obrigatório) O limite inferior para iteração de ponto fixo (`0` para consultas * ou `1` para consultas +).

Operador **`TermResolution`**

Converte valores de identificadores de strings internas de volta para suas strings externas correspondentes, ou converte strings externas em valores de identificadores de strings internas, dependendo do modo.

Modos

- `value2id`: mapeia termos, como literais e URIs, para valores de IDs internos correspondentes (codificação para valores internos).
- `id2value`: mapeia valores de IDs internos para os termos correspondentes, como literais e URIs, (decodificação de valores internos).

Argumentos

- `vars`: (obrigatório) especifica as variáveis cujas strings ou IDs de strings internas devem ser mapeados.

Operador **`Slice`**

Implementa um slice sobre o fluxo da solução de entrada, usando a semântica das cláusulas `LIMIT` e `OFFSET` do SPARQL.

Argumentos

- `limit`: (opcional) um limite das soluções a serem encaminhadas.
- `offset`: (opcional) o deslocamento em que as soluções são avaliadas para encaminhamento.

Operador **`SolutionInjection`**

Não recebe nenhuma entrada. Injeta estaticamente soluções no plano de consulta e registra-as no argumento `solutions`.

Os planos de consulta sempre começam com essa injeção estática. Se as soluções estáticas a serem injetadas puderem ser derivadas da própria consulta combinando várias origens de associações estáticas (por exemplo, de cláusulas `VALUES` ou `BIND`), o operador `SolutionInjection` injetará essas soluções estáticas derivadas. No caso mais simples, essas refletem associações que são implícitas por uma cláusula `VALUES` externa.

Se nenhuma solução estática puder ser derivada da consulta, `SolutionInjection` injetará a solução vazia, chamada de solução universal, que é expandida e multiplicada durante todo o processo de avaliação da consulta.

Argumentos

- `solutions`: (obrigatório) a sequência de soluções injetadas pelo operador.

Operador **`Sort`**

Classifica o conjunto de soluções usando as condições de classificação especificadas.

Argumentos

- `sortOrder`: (obrigatório) lista ordenada de variáveis, cada uma contendo um identificador `ASC` (crescente) ou `DESC` (decrescente), usado sequencialmente para classificar o conjunto de soluções.

Operador **`VariableAlignment`**

Inspeciona uma por uma das soluções, executando alinhamento em cada uma por meio de duas variáveis: uma `sourceVar` especificada e uma `targetVar` especificada.

Se `sourceVar` e `targetVar` em uma solução tiverem o mesmo valor, as variáveis serão consideradas alinhadas, e a solução será encaminhada, com a `sourceVar` redundante descartada.

Se as variáveis estiverem associadas a valores diferentes, a solução será filtrada por completo.

Argumentos

- `sourceVar`: (obrigatório) a variável de origem, a ser comparada com a variável de destino. Se o alinhamento for bem-sucedido em uma solução, indicando que as duas variáveis têm o mesmo valor, a variável de origem será descartada.
- `targetVar`: (obrigatório) a variável de destino, com a qual a variável de origem é comparada. É retida mesmo quando o alinhamento é bem-sucedido.

Limitações do SPARQL **`explain`** no Neptune

A versão do atributo `explain` do SPARQL no Neptune tem as limitações a seguir.

No momento, o Neptune é compatível com explain apenas em consultas SELECT do SPARQL

Para obter informações sobre o processo de avaliação para outras formas de consulta, como consultas `ASK`, `CONSTRUCT`, `DESCRIBE` e `SPARQL UPDATE`, você pode transformar essas consultas em uma consulta SELECT. Depois, use `explain` para inspecionar a consulta SELECT correspondente.

Por exemplo, para obter informações do `explain` sobre uma consulta `ASK WHERE {...}`, execute a consulta `SELECT WHERE {...} LIMIT 1` correspondente com o `explain`.

Da mesma forma, em uma consulta `CONSTRUCT {...} WHERE {...}`, descarte a parte `CONSTRUCT {...}` e execute uma consulta `SELECT` com `explain` na segunda cláusula `WHERE {...}`. A avaliação da segunda cláusula `WHERE`, geralmente revela os principais desafios do processamento da consulta `CONSTRUCT`, pois as soluções do fluxo de saída da segunda `WHERE` no modelo `CONSTRUCT` geralmente exigem apenas substituição direta.

Os operadores do explain podem ser alterados em versões futuras

Os operadores e parâmetros do `explain` do SPARQL podem ser alterados em versões futuras.

A saída do explain pode ser alterada em versões futuras

Por exemplo, os cabeçalhos das colunas podem ser alterados e mais colunas podem ser adicionadas às tabelas.

Consultas federadas do SPARQL no Neptune usando a extensão `SERVICE`

O Amazon Neptune é totalmente compatível com a extensão de consultas federadas do SPARQL que usa a palavra-chave `SERVICE`. (Para obter mais informações, consulte SPARQL 1.1 Federated Query.)

> **Note**
>
> Esse atributo está disponível a partir da Versão 1.0.1.0.200463.0 (15/10/2019).

A palavra-chave `SERVICE` instrui o mecanismo de consulta do SPARQL a executar uma parte da consulta em um endpoint remoto do SPARQL e compor o resultado da consulta final. Somente operações `READ` são possíveis. As operações `WRITE` e `DELETE` não são compatíveis. O Neptune só pode executar consultas federadas em endpoints SPARQL acessíveis em sua nuvem privada virtual (VPC). No entanto, também é possível usar um proxy reverso na VPC para tornar uma fonte de dados externa acessível dentro da VPC.

> **Note**
>
> Quando o `SERVICE` do SPARQL é usado para federar uma consulta em dois ou mais clusters do Neptune na mesma VPC, os grupos de segurança devem ser configurados para permitir que todos esses clusters do Neptune se comuniquem uns com os outros.

> **Important**
>
> A federação do SPARQL 1.1 faz solicitações de serviço em seu nome ao transmitir consultas e parâmetros para endpoints do SPARQL externos. É sua responsabilidade verificar se os endpoints do SPARQL externos atendem aos requisitos de segurança e tratamento de dados de seu aplicativo.

Exemplo de consulta federada do Neptune

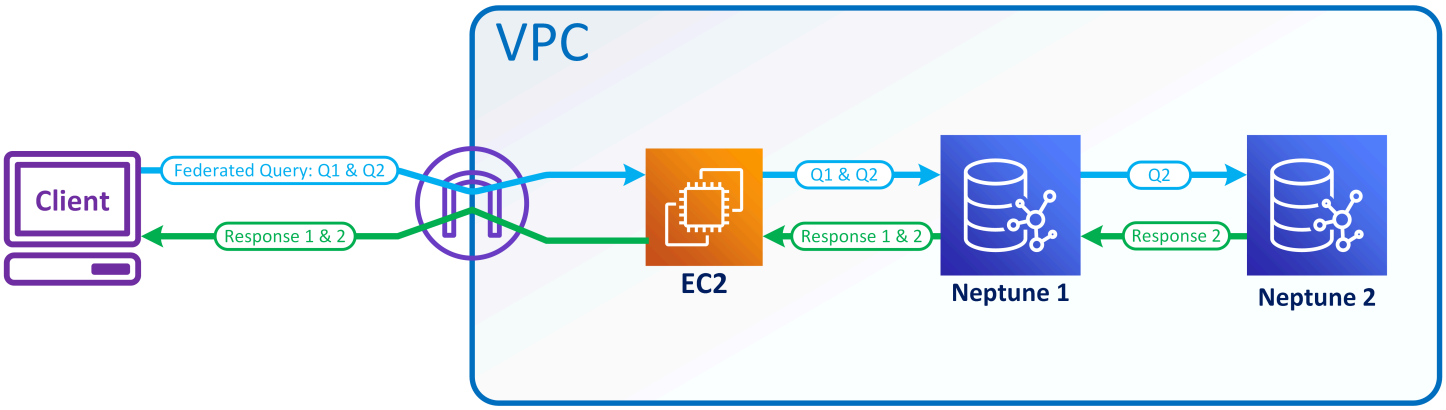
O exemplo simples a seguir mostra como funcionam as consultas federadas do SPARQL.

Suponha que um cliente envie a seguinte consulta ao Neptune-1 em `http://neptune-1:8182/sparql`.

```
SELECT * WHERE {
   ?person rdf:type foaf:Person .
   SERVICE <http://neptune-2:8182/sparql> {
       ?person foaf:knows ?friend .
    }
}
```

1. O Neptune-1 avalia o primeiro padrão de consulta (Q-1) que é `?person rdf:type foaf:Person`, usa os resultados para resolver `?person` em Q-2 (`?person foaf:knows ?friend`) e encaminha o padrão resultante para o Neptune-2 em `http://neptune-2:8182/sparql`.
2. O Neptune-2 avalia Q-2 e envia os resultados de volta ao Neptune-1.
3. Neptune-1 une as soluções dos dois padrões e envia os resultados de volta ao cliente.

Esse fluxo é mostrado no diagrama a seguir.



> **Note**
>
> “Por padrão, o otimizador determina em que ponto da execução da consulta a instrução `SERVICE` é executada. É possível substituir esse posicionamento usando a dica de consulta joinOrder.

Controle de acesso de consultas federadas no Neptune

O Neptune AWS Identity and Access Management usa (IAM) para autenticação e autorização. O controle de acesso de uma consulta federada pode envolver mais de uma instância de banco de dados do Neptune. Essas instâncias podem ter requisitos diferentes de controle de acesso. Em determinadas circunstâncias, isso pode limitar sua capacidade de fazer uma consulta federada.

Considere o exemplo simples apresentado na seção anterior. O Neptune-1 chama o Neptune-2 com as mesmas credenciais com as quais ele foi chamado.

- Se o Neptune-1 exigir autenticação e autorização do IAM, mas o Neptune-2 não exigir, bastará ter as permissões apropriadas do IAM para o Neptune-1 para fazer a consulta federada.
- Se o Neptune-1 e o Neptune-2 exigirem a autenticação e a autorização do IAM, você precisará associar as permissões do IAM para os dois bancos de dados para fazer a consulta federada. Os dois clusters também devem estar na mesma AWS conta e na mesma região. Atualmente, não há suporte para arquiteturas de consulta federada entre regiões e/ou entre contas.
- No entanto, quando o Neptune-1 não estiver habilitado para o Neptune-2, você não poderá fazer uma consulta federada. O motivo é que o Neptune-1 não pode recuperar suas credenciais do IAM e transmiti-las ao Neptune-2 para autorizar a segunda parte da consulta.

Ferramentas de visualização de grafos para Neptune

Além dos recursos de visualização incorporados aos cadernos gráficos Neptune, você também pode usar soluções criadas AWS por parceiros e fornecedores terceirizados para visualizar dados armazenados no Neptune.

A visualização sofisticada de grafos pode ajudar cientistas de dados, gerentes e outras funções em uma organização a explorar dados de grafos de forma interativa, sem precisar saber como escrever consultas complexas.

Tópicos

- O graph-explorer de código aberto
- Tom Sawyer Software
- Cambridge Intelligence
- Graphistry
- metaphacts
- G.V( )
- Linkurious

O graph-explorer de código aberto

O graph-explorer é uma ferramenta de exploração visual de código aberto para dados de grafos com pouco código, disponível sob a licença Apache-2.0. Ele permite percorrer dados de grafos de propriedades rotulados (LPG) ou do Resource Description Framework (RDF) em um banco de dados de grafos sem precisar escrever consultas de grafo. O graph-explorer tem como objetivo ajudar cientistas de dados, analistas de negócios e outras funções em uma organização a explorar dados de grafos de forma interativa sem precisar aprender uma linguagem de consulta de grafos.

O graph-explorer oferece uma aplicação web baseada em React que pode ser implantada como um contêiner para visualizar dados de grafos. Você pode se conectar ao Amazon Neptune ou a outros bancos de dados gráficos que fornecem um endpoint TinkerPop Apache Gremlin ou SPARQL 1.1.

- É possível ver rapidamente um resumo dos dados usando os filtros facetados ou pesquisar os dados digitando texto na barra de pesquisa.

- Você também pode examinar de forma interativa as conexões de nós e bordas. Você pode visualizar os vizinhos dos nós para ver como os objetos se relacionam entre si e, depois, detalhar para inspecionar visualmente as bordas e as propriedades.
- Você também pode personalizar o layout do grafo, as cores, os ícones e as propriedades padrão a serem exibidas para nós e bordas. Para grafos do RDF, você também pode personalizar namespaces para URIs de recursos.
- Para relatórios e apresentações que envolvam dados de grafos, é possível configurar e salvar visualizações criadas em um formato PNG de alta resolução. Você também pode baixar os dados associados em um arquivo CSV ou JSON para processamento adicional.

Usar o graph-explorer em um bloco de anotações do Neptune

A maneira mais fácil de usar o graph-explorer com o Neptune é em um bloco de anotações do Neptune.

Se você usar a bancada de trabalho do Neptune para hospedar um bloco de anotações do Neptune, o graph-explorer será automaticamente implantado com o bloco de anotações e conectado ao Neptune.

Depois de criar um bloco de anotações, acesse o console do Neptune para iniciar o graph-explorer:

1. Vá para Neptune.
2. Em Blocos de anotações, selecione o seu.
3. Em Ações, escolha Abrir o Graph Explorer.

Como executar o graph-explorer no Amazon ECS em AWS Fargate e se conectar ao Neptune

Você também pode criar a imagem Docker do explorador de gráficos e executá-la em uma máquina local ou em um serviço hospedado, como o Amazon Elastic Compute Cloud (Amazon EC2) ou o Amazon ElasticContainer Service (Amazon ECS), conforme explicado na seção Getting Started do read-me no projeto graph-explorer. GitHub

Como exemplo, esta seção fornece step-by-step instruções para executar o explorador de gráficos no Amazon ECS em: AWS Fargate

1. Crie um perfil do IAM e associe a ele estas políticas:

- AmazonECS TaskExecution RolePolicy
- CloudWatchLogsFullAcesso

Mantenha o nome do perfil à mão para usá-lo em breve.

2. Crie um cluster do Amazon ECS com a infraestrutura definida como FARGATE e as seguintes opções de rede:

   - VPC: defina como a VPC em que seu banco de dados Neptune está localizado.
   - Subnets: defina como as sub-redes públicas dessa VPC (remova todas as outras).

3. Crie uma definição de tarefa JSON da seguinte forma:

```
{
  "family": "explorer-test",
  "containerDefinitions": [
    {
      "name": "graph-explorer",
      "image": "public.ecr.aws/neptune/graph-explorer:latest",
      "cpu": 0,
      "portMappings": [
        {
          "name": "graph-explorer-80-tcp",
          "containerPort": 80,
          "hostPort": 80,
          "protocol": "tcp",
          "appProtocol": "http"
        },
        {
          "name": "graph-explorer-443-tcp",
          "containerPort": 443,
          "hostPort": 443,
          "protocol": "tcp",
          "appProtocol": "http"
        }
      ],
      "essential": true,
      "environment": [
        {
          "name": "HOST",
          "value": "localhost"
        }
```

```
      ],
      "mountPoints": [],
      "volumesFrom": [],
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-create-group": "true",
          "awslogs-group": "/ecs/graph-explorer",
          "awslogs-region": "{region}",
          "awslogs-stream-prefix": "ecs"
        }
      }
    }
  ],
  "taskRoleArn": "arn:aws:iam::{account_no}:role/{role_name_from_step_1}",
  "executionRoleArn": "arn:aws:iam::{account_no}:role/{role_name_from_step_1}",
  "networkMode": "awsvpc",
  "requiresCompatibilities": [
    "FARGATE"
  ],
  "cpu": "1024",
  "memory": "3072",
  "runtimePlatform": {
    "cpuArchitecture": "X86_64",
    "operatingSystemFamily": "LINUX"
  }
}
```

4. Inicie uma nova tarefa usando as configurações padrão, exceto nos seguintes campos:

   - Ambiente
     - Opções de computação => Tipo de inicialização
   - Configuração de implantação
     - Tipo de aplicativo => Tarefa
     - Família => *(sua nova definição de tarefa JSON*)
     - Revisão => *(mais recente)*
   - Redes
     - VPC => (*a VPC do Neptune à qual você deseja se conectar)*
     - Sub-redes => *(SOMENTE as sub-redes públicas da VPC: remova todas as outras)*

- Grupo de segurança => Criar um novo grupo de segurança
- Nome do grupo de segurança => graph-explorer
- Descrição do grupo de segurança = grupo de segurança para acesso ao graph-explorer
- Regras de entrada para um grupo de segurança =>
  1. 80 Anywhere
  2. 443 Anywhere

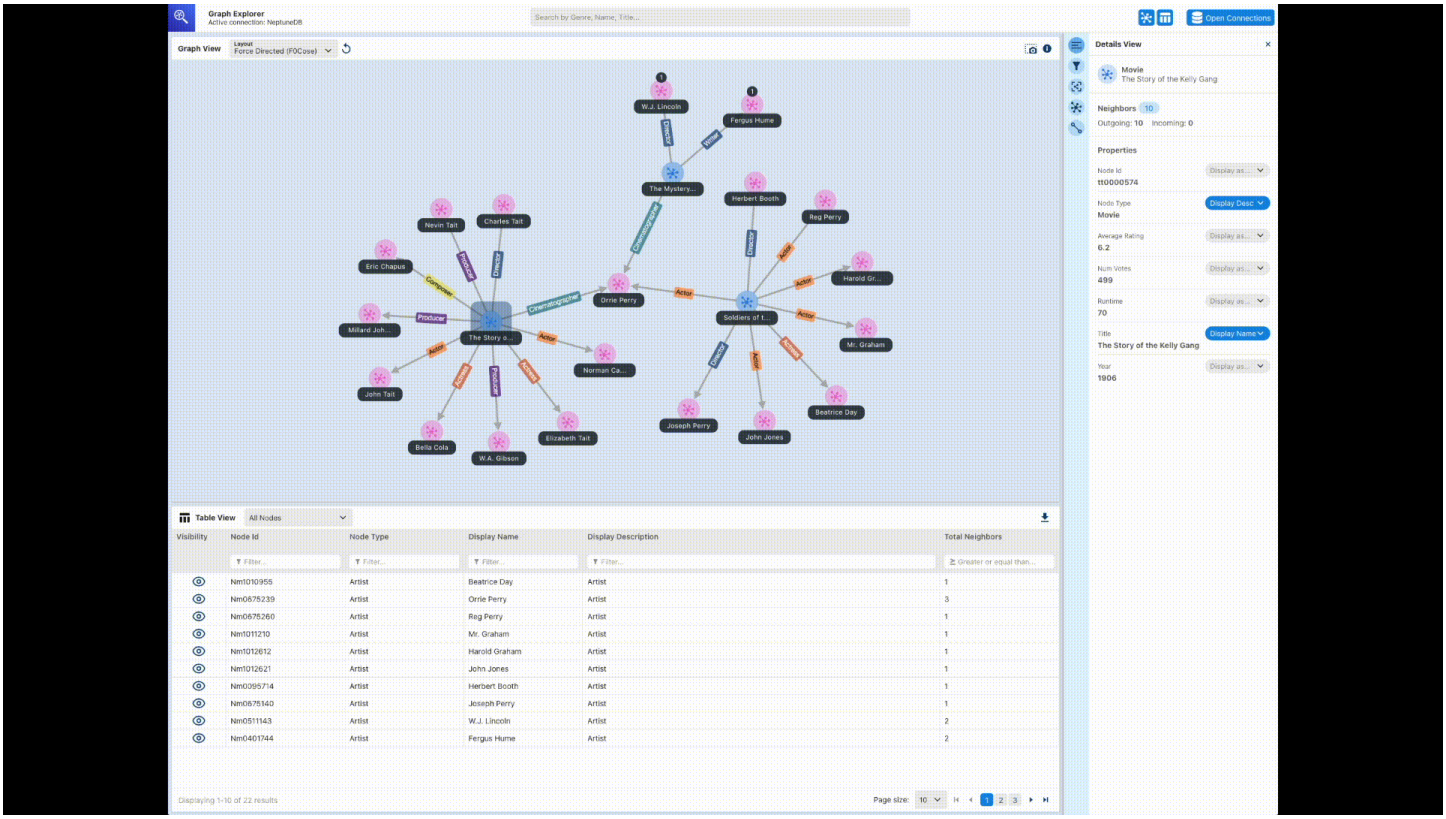
5. Selecione Criar.
6. Depois que a tarefa começar, copie o IP público da tarefa em execução e navegue até: `https://(your public IP)/explorer`.
7. Aceite o risco de usar o certificado não reconhecido que foi gerado ou adicione-o à cadeia de chaves.
8. Agora você pode adicionar uma conexão com o Neptune. Crie uma conexão, seja para um grafo de propriedades (LPG) ou para RDF, e defina os seguintes campos:

```
Using proxy server => true
Public or Proxy Endpoint => https://(your public IP address)
Graph connection URL => https://(your Neptune endpoint):8182
```

Agora você deve estar conectado.

Demonstração do graph-explorer

Este breve vídeo dá uma ideia de como visualizar com facilidade os dados de grafos usando o graph-explorer:



Tom Sawyer Software

Tom Sawyer Perspectives é uma plataforma de desenvolvimento de análise e visualização de dados e grafos de pouco código para dados armazenados no Amazon Neptune. As interfaces integradas de design e visualização e as extensas bibliotecas de API permitem que você crie com rapidez aplicações de visualização personalizadas e com qualidade de produção. Com uma interface de point-and-click designer e 30 algoritmos de análise integrados, você pode projetar e desenvolver aplicativos para obter insights sobre dados agrupados de dezenas de fontes.

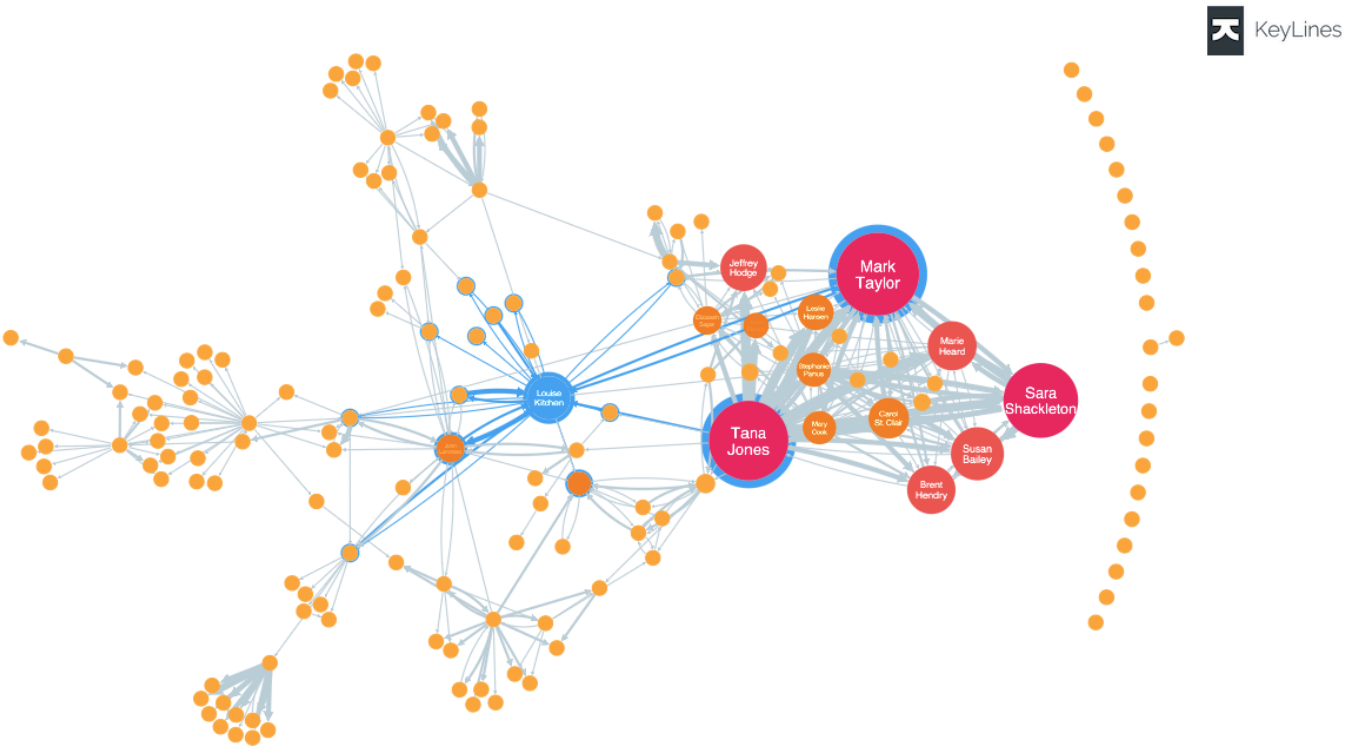
O Tom Sawyer Graph Database Browser facilita a visualização e a análise de dados no Amazon Neptune. Você pode ver e entender as conexões nos dados sem ter um amplo conhecimento da linguagem nem do esquema de consulta. É possível interagir com os dados sem conhecimento técnico simplesmente carregando os vizinhos dos nós selecionados e criando a visualização em qualquer direção necessária. Você também pode utilizar cinco layouts de grafo exclusivos para exibir o grafo de uma forma que forneça o máximo de significado e aplicar análises de centralidade, agrupamento e busca de caminhos para revelar padrões nunca vistos anteriormente. Para ver um exemplo da integração do Graph Database Browser com o Neptune, confira esta postagem no blog. Para começar com uma avaliação gratuita do Graph Database Browser, acesse o AWS Marketplace.

Cambridge Intelligence

A Cambridge Intelligence oferece tecnologias de visualização de dados para examinar e entender os dados do Amazon Neptune. Os kits de ferramentas de visualização gráfica (KeyLinespara JavaScript desenvolvedores e ReGraphdesenvolvedores do React) oferecem uma maneira fácil de criar ferramentas altamente interativas e personalizáveis para aplicativos web. Esses kits de ferramentas utilizam o WebGL e o HTML5 Canvas para um desempenho rápido, são compatíveis com funções avançadas de análise de grafos e combinam flexibilidade e escalabilidade com uma arquitetura segura e sólida. Esses SDKs funcionam com dados do Gremlin e do RDF no Neptune.

Confira estes tutoriais de integração para dados do Gremlin dados do SPARQL e arquitetura do Neptune.

Aqui está um exemplo de KeyLines visualização:

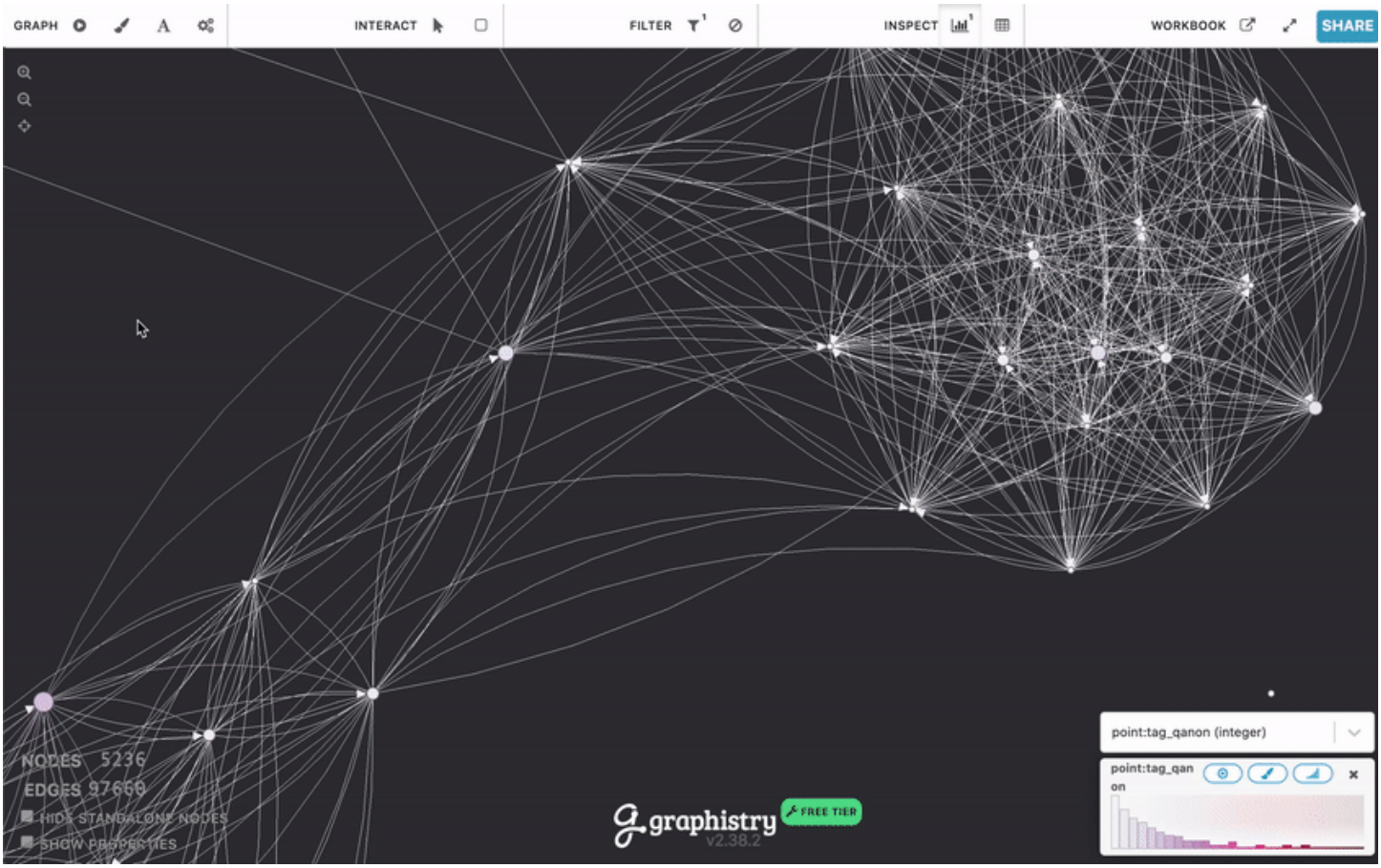


Graphistry

Graphistry é uma plataforma de inteligência gráfica visual que aproveita a aceleração da GPU para experiências visuais ricas. As equipes podem colaborar no Graphistry usando uma série de atributos, desde a exploração sem código de arquivos e bancos de dados, o compartilhamento de cadernos Jupyter e painéis do Streamlit até o uso da API de incorporação nas próprias aplicações.

É possível começar com painéis totalmente interativos de pouca codificação simplesmente configurando e iniciando o graph-app-kit e modificando apenas algumas linhas de código. Confira esta postagem no blog para ver um passo a passo sobre como criar o primeiro painel usando o Graphistry e o Neptune. Você também pode experimentar a demonstração do PyGraphistryNeptune. PyGraphistry é uma biblioteca de análise gráfica visual em Python para notebooks. Confira este caderno tutorial para ver uma demonstração do PyGraphistry Neptune.

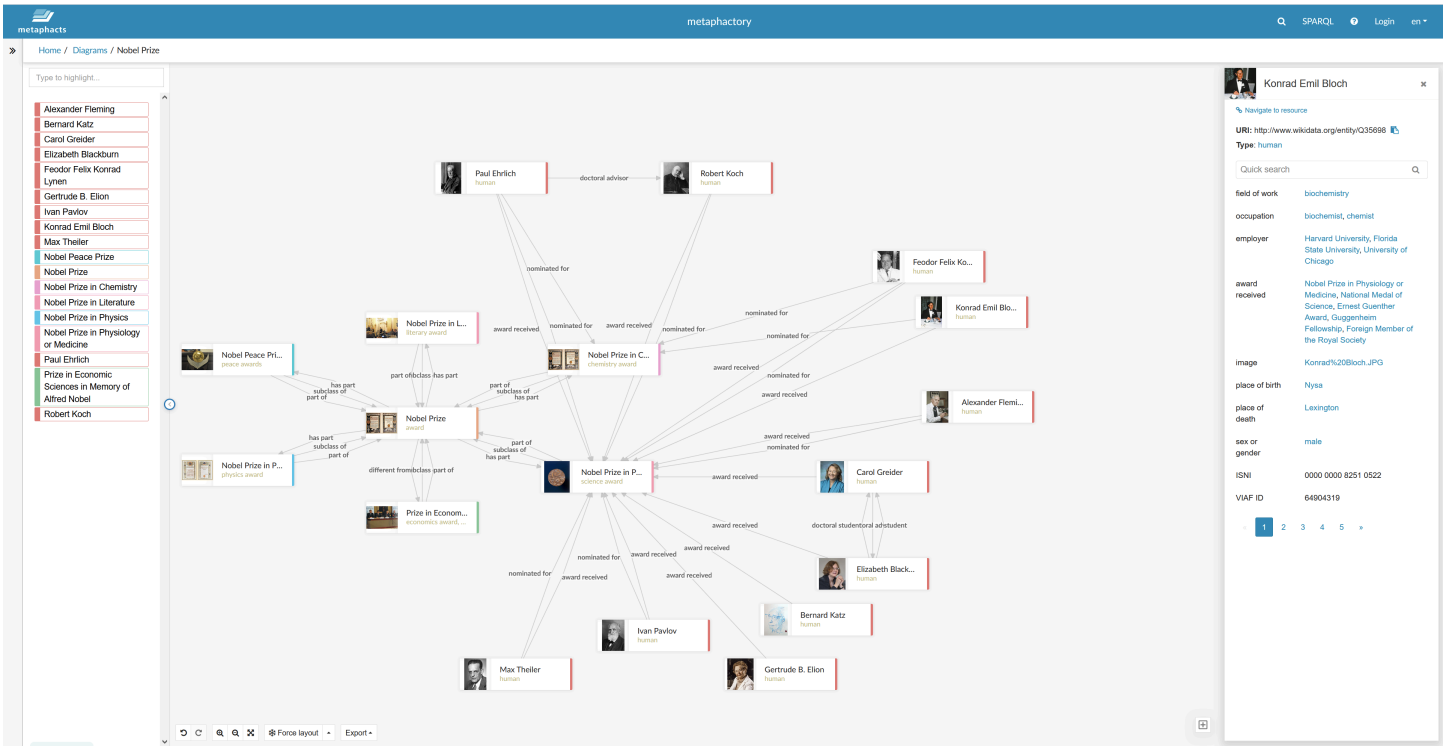
Para começar, acesse Graphistry no Marketplace AWS.



metaphacts

O metaphacts oferece uma plataforma flexível e aberta para descrever e consultar dados de grafos, bem como visualizar e interagir com grafos de conhecimento. Usando a metaphactory, é possível criar aplicações web interativas, como visualizações e painéis sobre grafos de conhecimento no Neptune, usando o modelo de dados do RDF. A plataforma metaphactory proporciona uma experiência de desenvolvimento de pouco código com uma interface de usuário para carregamento de dados, uma interface visual de modelagem de ontologia com suporte a OWL e SHACL, uma interface de consulta e catálogo de consultas SPARQL e um rico conjunto de componentes da web para exploração, visualização, pesquisa e criação de grafos.

Veja um exemplo de visualização da metaphactory:



A plataforma foi projetada e usada de forma produtiva em engenharia, fabricação, farmacêutica, ciências biológicas, finanças, seguros e muito mais. Para ver um exemplo de arquitetura de solução, confira esta postagem no blog.

Para começar com uma avaliação gratuita da metaphactory, acesse o AWS Marketplace.

G.V( )

G.V () é uma poderosa ferramenta Gremlin Integrated Development Environment (IDE) para desenvolvedores e analistas de dados. Com ele, você pode consultar, visualizar e atualizar dados de grafos de forma interativa no Neptune. G.V () oferece a funcionalidade integrada de preenchimento automático da linguagem Gremlin, que fornece sugestões e documentação à medida que você digita sua consulta, com base em seu modelo de dados gráficos.

Você também pode usar o atributo de depuração de consultas do Gremlin para escrever, depurar, testar e analisar os processos de percursos de grafos em profundidade.

Com o Processamento de Linguagem Natural desenvolvido pela OpenAI, G.V () pode gerar consultas Gremlin precisas para seu esquema de dados gráficos a partir de um prompt de texto, para consultar seus dados por meio de linguagem natural.

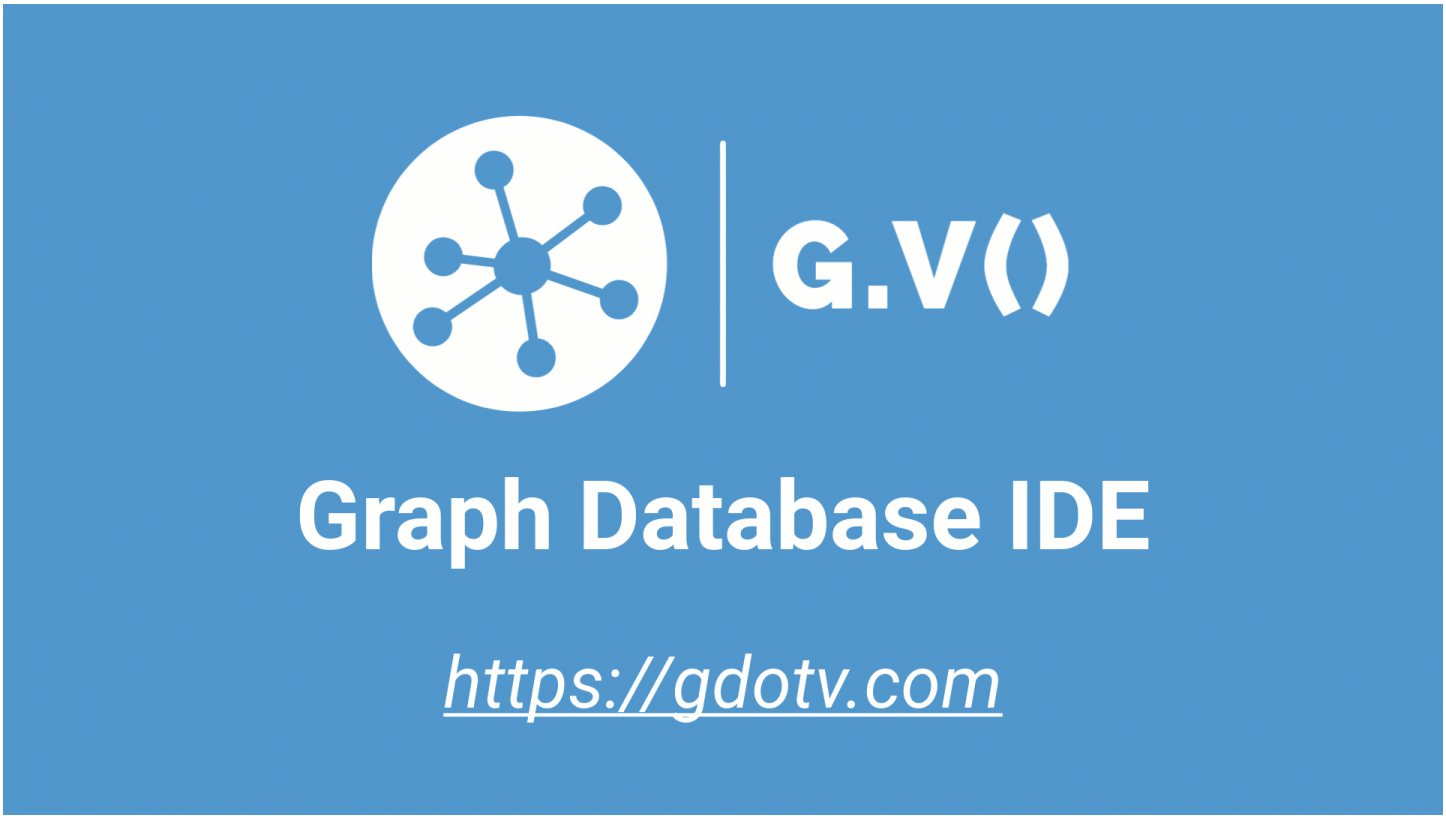
O Graph Data Explorer permite que você navegue e modifique seu gráfico para arquitetar rapidamente novas estruturas gráficas e manter as existentes.

G.V () oferece vários formatos de visualização para resultados de consultas que ajudam você a interpretar a saída da consulta e a navegar pelo gráfico de forma interativa. Isso inclui formatos de saída de console de tabela, grafo, JSON e Gremlin.

O G.V () é totalmente compatível com o Amazon Neptune e oferece muitos recursos adicionais específicos para o Amazon Neptune, como insights do Slow Query ou do Audit Log e suporte à autenticação do IAM. Para saber mais, confira a documentação.

G.V () está em constante evolução e recebe novos recursos mensalmente. Para saber mais sobre G.V (), comece com um teste gratuito visitando o site G.V ().

Veja abaixo uma demonstração do G.V () em ação:



Linkurious

O Linkurious fornece diferentes soluções de inteligência gráfica para usuários técnicos e não técnicos e uma variedade de casos de uso.

O Linkurious Enterprise Explorer é um software de visualização e análise de off-the-shelf gráficos criado para equipes que podem acompanhar as demandas de suas day-to-day atividades e ajudar os profissionais orientados por dados a fazer grandes coisas de forma simples. Totalmente configurável e fácil de usar, ele se adapta facilmente às suas necessidades e capacita usuários novatos ou avançados a visualizar dados rapidamente em AWS Neptune, a explorar intuitivamente seu conjunto de dados, independentemente do tamanho ou da complexidade dos dados, e a colaborar perfeitamente nos níveis de equipe ou empresa.

O Linkurious Enterprise Watchtower aproveita o poder do Linkurious Enterprise Explorer e adiciona recursos inovadores de detecção e gerenciamento de casos para oferecer um software integrado de detecção e investigação com tecnologia gráfica. Por um lado, ele permite que você configure alertas que utilizam o Neptune Database e o Neptune Analytics para revelar automaticamente anomalias ou padrões em dados conectados complexos. Por outro lado, ele combina recursos de gerenciamento de casos e colaboração para ajudar as equipes a gerenciar com eficiência seus fluxos de trabalho investigativos.

O Ogma é uma JavaScript biblioteca comercial que ajuda você a desenvolver visualizações gráficas interativas poderosas e em grande escala para seus aplicativos. Ele aproveita a renderização WebGL e os layouts de alto desempenho para permitir que os usuários exibam e interajam com milhares de nós e bordas em questão de segundos. Ele também fornece uma variedade de recursos para personalizar seu aplicativo e criar experiências de usuário ricas. Por fim, ele vem equipado com documentação e ferramentas abrangentes, como tutoriais, dezenas de exemplos e um playground interativo.

Para começar, solicite um teste gratuito de 30 dias do Linkurious Enterprise ou Ogma.

Exportar dados de um cluster de banco de dados do Neptune

Há algumas maneiras boas de exportar dados de um cluster de banco de dados do Neptune:

- Para pequenas quantidades de dados, basta usar os resultados de uma consulta ou consultas.
- Para dados do RDF, o Graph Store Protocol (GSP) pode facilitar a exportação. Por exemplo:

```
curl --request GET \
  'https://your-neptune-endpoint:port/sparql/gsp/?graph=http%3A//www.example.com/named/graph'
```

- Há também uma ferramenta de código aberto avançada e flexível para exportar dados do Neptune, ou seja, `neptune-export`. As seções a seguir descrevem como usar os atributos desta ferramenta e como usá-la.

Tópicos

Usar o **neptune-export**

É possível utilizar a ferramenta `neptune-export` de código aberto de duas formas diferentes:

- Como o serviço Neptune-Export. Ao exportar dados do Neptune usando o serviço Neptune-Export, você aciona e monitora trabalhos de exportação por meio de uma API REST.
- Como o utilitário de linha de comando Java **`neptune-export`**. Para usar essa ferramenta de linha de comando para exportar dados do Neptune, é necessário executá-la em um ambiente em que o cluster de banco de dados do Neptune esteja acessível.

Tanto o serviço Neptune-Export quanto a ferramenta de linha de comando `neptune-export` publicam dados no Amazon Simple Storage Service (Amazon S3), criptografados usando criptografia no lado do servidor Amazon S3 (`SSE-S3`).

> **Note**
>
> É uma prática recomendada habilitar o registro em log do acesso em todos os buckets do Amazon S3, para permitir que você audite todo o acesso a esses buckets.

Se você tentar exportar dados de um cluster de banco de dados do Neptune cujos dados estejam mudando enquanto a exportação está acontecendo, a consistência dos dados exportados não será garantida. Ou seja, se o cluster estiver atendendo ao tráfego de gravação enquanto um trabalho de exportação está em andamento, poderá haver inconsistências nos dados exportados. Esse será o caso se você fizer a exportação da instância principal no cluster ou de uma ou mais réplicas de leitura.

Para garantir que os dados exportados sejam consistentes, é melhor exportar de um clone do cluster de banco de dados. Isso oferece à ferramenta de exportação uma versão estática dos dados e garante que o trabalho de exportação não diminua a velocidade das consultas no cluster de banco de dados original.

Para facilitar isso, é possível indicar que deseja clonar o cluster de banco de dados de origem ao acionar um trabalho de exportação. Se você fizer isso, o processo de exportação criará automaticamente o clone, o usará para a exportação e, depois, o excluirá quando a exportação for concluída.

Usar o serviço Neptune-Export para exportar dados do Neptune

É possível usar as etapas a seguir para exportar dados do cluster de banco de dados do Neptune para o Amazon S3 usando o serviço Neptune-Export:

Instalar o serviço Neptune-Export

Use um modelo do AWS CloudFormation para criar a pilha:

Como instalar o serviço Neptune-Export

1. Inicie a pilha do AWS CloudFormation no console do AWS CloudFormation selecionando um dos botões Iniciar pilha na seguinte tabela:

Região	Visualização	Visualizar no Designer	Executar
Leste dos EUA (Norte da Virgínia)	Visão	Visualizar no Designer	
Leste dos EUA (Ohio)	Visão	Visualizar no Designer	
Oeste dos EUA (N. da Califórnia)	Visão	Visualizar no Designer	
Oeste dos EUA (Oregon)	Visão	Visualizar no Designer	
Canadá (Central)	Visão	Visualizar no Designer	
América do Sul (São Paulo)	Visão	Visualizar no Designer	
Europa (Estocolmo)	Visão	Visualizar no Designer	

Região	Visualização	Visualizar no Designer	Executar
Europa (Irlanda)	Visão	Visualizar no Designer	
Europa (Londres)	Visão	Visualizar no Designer	
Europa (Paris)	Visão	Visualizar no Designer	
Europa (Frankfurt)	Visão	Visualizar no Designer	
Oriente Médio (Barém)	Visão	Visualizar no Designer	
Oriente Médio (Emirados Árabes Unidos)	Visão	Visualizar no Designer	
Israel (Tel Aviv)	Visão	Visualizar no Designer	
África (Cidade do Cabo)	Visão	Visualizar no Designer	
Ásia-Pacífico (Hong Kong)	Visão	Visualizar no Designer	
Ásia-Pacífico (Tóquio)	Visão	Visualizar no Designer	
Ásia-Pacífico (Seul)	Visão	Visualizar no Designer	
Ásia-Pacífico (Singapura)	Visão	Visualizar no Designer	

Região	Visualização	Visualizar no Designer	Executar
Ásia-Pacífico (Sydney)	Visão	Visualizar no Designer	
Ásia-Pacífico (Mumbai)	Visão	Visualizar no Designer	
China (Pequim)	Visão	Visualizar no Designer	
China (Ningxia)	Visão	Visualizar no Designer	
AWS GovCloud (Oeste dos EUA)	Visão	Visualizar no Designer	
AWS GovCloud (Leste dos EUA)	Visão	Visualizar no Designer	

2. Na página Select Template, escolha Next.
3. Na página Especificar detalhes, defina os seguintes parâmetros:

   - **`VPC`**: a maneira mais fácil de configurar o serviço Neptune-Export é instalá-lo na mesma Amazon VPC do banco de dados Neptune. Se quiser instalá-lo em uma VPC separada, você poderá usar o emparelhamento da VPC para estabelecer conectividade entre a VPC do cluster de banco de dados do Neptune e a VPC do serviço Neptune-Export.
   - **`Subnet1`**: o serviço Neptune-Export deve ser instalado em uma sub-rede na VPC que permita tráfego HTTPS IPv4 de saída da sub-rede para a Internet. Isso é para que o serviço Neptune-Export possa chamar a API AWS Batch para criar e executar um trabalho de exportação.

     Se você criou o cluster do Neptune usando o modelo do CloudFormation na página Criar um cluster de banco de dados da documentação do Neptune, poderá usar as saídas `PrivateSubnet1` e `PrivateSubnet2` dessa pilha para preencher esse e o próximo parâmetro.
   - **`Subnet2`**: a segunda sub-rede na VPC que viabiliza tráfego HTTPS IPv4 de saída da sub-rede para a Internet.

- **`EnableIAM`**: defina como `true` para proteger a API Neptune-Endpoint usando AWS Identity and Access Management (IAM). Recomendamos que você o faça.

  Se você habilitar a autenticação do IAM, deverá assinar com `Sigv4` todas as solicitações HTTPS no endpoint. É possível usar uma ferramenta, como o awscurl, para assinar solicitações em seu nome.

- **`VPCOnly`**: definir como `true` torna o endpoint de exportação somente para VPC, para que você só possa acessá-lo de dentro da VPC em que o serviço Neptune-Export está instalado. Isso restringe a API Neptune-Export para ser usada somente de dentro dessa VPC.

  Recomendamos que você defina `VPCOnly` como `true`.

- **`NumOfFilesULimit`** : especifique um valor entre dez mil e um milhão para `nofile` na propriedade do contêiner `ulimits`. O padrão é dez mil, e recomendamos manter o padrão, a menos que o grafo contenha um grande número de rótulos exclusivos.

- **`PrivateDnsEnabled`** (booliano): indica se você deseja ou não associar uma zona hospedada privada à VPC especificada. O valor padrão é `true`.

  Quando um endpoint da VPC é criado com esse sinalizador habilitado, todo o tráfego do API Gateway é direcionado pelo endpoint da VPC, e as chamadas públicas do endpoint do API Gateway são desabilitadas. Se você definir `PrivateDnsEnabled` como `false`, o endpoint público do API Gateway será habilitado, mas o serviço de exportação do Neptune não poderá ser conectado por meio do endpoint de DNS privado. Depois, é possível usar um endpoint do DNS público para que o endpoint da VPC chame o serviço de exportação, conforme detalhado aqui.

4. Escolha Next (Próximo).
5. Na página Options (Opções), escolha Next (Avançar).
6. Na página Revisar, marque a primeira caixa de seleção para confirmar que o AWS CloudFormation criará recursos do IAM. Marque a segunda caixa de seleção para confirmar `CAPABILITY_AUTO_EXPAND` para a nova pilha.

> **Note**
>
> `CAPABILITY_AUTO_EXPAND` confirma explicitamente que os macros serão expandidos ao criar a pilha, sem revisão anterior. Os usuários geralmente criam um conjunto de alterações a partir de um modelo processado para que as alterações feitas pelos macros

possam ser revisadas antes de criar a pilha. Para obter mais informações, consulte a API CreateStack do AWS CloudFormation.

Em seguida, selecione Criar.

Habilitar o acesso ao Neptune por meio do Neptune-Export

Depois que a instalação do Neptune-Export for concluída, atualize o grupo de segurança da VPC do Neptune para permitir o acesso do Neptune-Export. Quando a pilha do AWS CloudFormation do Neptune-Export tiver sido criada, a guia Saídas incluirá um ID `NeptuneExportSecurityGroup`. Atualize o grupo de segurança da VPC do Neptune para permitir o acesso desse grupo de segurança do Neptune-Export.

Habilitar o acesso ao endpoint do Neptune-Export por meio de uma instância do EC2 baseada em VPC

Se você tornar o endpoint do Neptune-Export somente para VPC, só poderá acessá-lo de dentro da VPC em que o serviço Neptune-Export estiver instalado. Para viabilizar a conectividade de uma instância do Amazon EC2 na VPC a partir da qual você pode fazer chamadas à API do Neptune-Export, anexe o `NeptuneExportSecurityGroup` criado pela pilha AWS CloudFormation a essa instância do Amazon EC2.

Executar um trabalho do Neptune-Export usando a API do Neptune-Export

A guia Saídas da pilha AWS CloudFormation também inclui o `NeptuneExportApiUri`. Use esse URI sempre que enviar uma solicitação ao endpoint do Neptune-Export.

Executar um trabalho de exportação

- Assegure-se de que o usuário ou o perfil sob o qual a exportação é executada tenha recebido a permissão `execute-api:Invoke`.
- Se você definir o parâmetro `EnableIAM` como `true` na pilha AWS CloudFormation ao instalar o Neptune-Export, precisará assinar com `Sigv4` todas as solicitações na API do Neptune-Export. Recomendamos usar o awscurl para fazer solicitações à API. Todos os exemplos aqui pressupõem que a autenticação do IAM esteja habilitada.

- Se você definir o parâmetro `VPCOnly` como `true` na pilha AWS CloudFormation ao instalar o Neptune-Export, deverá chamar a API do Neptune-Export de dentro da VPC, normalmente de uma instância do Amazon EC2 localizada na VPC.

Para começar a exportar dados, envie uma solicitação ao endpoint `NeptuneExportApiUri` com os parâmetros de solicitação `command` e `outputS3Path` e um parâmetro de exportação `endpoint`.

Veja um exemplo de solicitação que exporta dados de grafos de propriedades do Neptune e os publica no Amazon S3:

```
curl \
  (your NeptuneExportApiUri) \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "command": "export-pg",
        "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
        "params": { "endpoint": "(your Neptune endpoint DNS name)" }
      }'
```

Da mesma forma, veja um exemplo de solicitação que exporta dados do RDF do Neptune para o Amazon S3:

```
curl \
  (your NeptuneExportApiUri) \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "command": "export-rdf",
        "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
        "params": { "endpoint": "(your Neptune endpoint DNS name)" }
      }'
```

Se você omitir o parâmetro de solicitação `command`, por padrão, o Neptune-Export tentará exportar dados do grafo de propriedades do Neptune.

Se o comando anterior fosse executado com êxito, a saída ficaria desta forma:

```
{
  "jobName": "neptune-export-abc12345-1589808577790",
```

```
  "jobId": "c86258f7-a9c9-4f8c-8f4c-bbfe76d51c8f"
}
```

Monitorar o trabalho de exportação que você acabou de iniciar

Para monitorar um trabalho em execução, anexe o jobID ao `NeptuneExportApiUri`, mais ou menos desta forma:

```
curl \
  (your NeptuneExportApiUri)(the job ID)
```

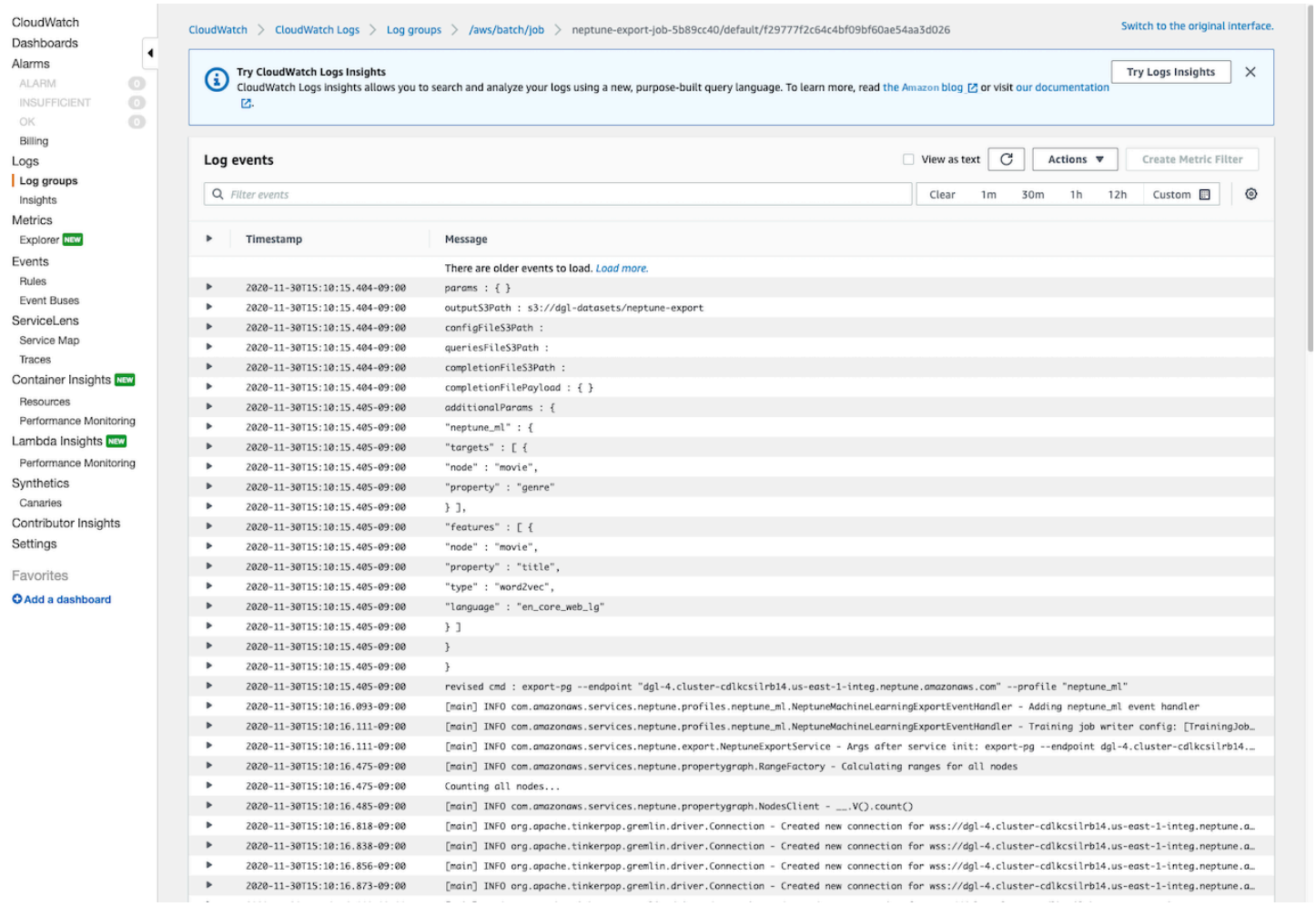
Se o serviço ainda não tivesse iniciado o trabalho de exportação, a resposta ficaria desta forma:

```
{
  "jobId": "c86258f7-a9c9-4f8c-8f4c-bbfe76d51c8f",
  "status": "pending"
}
```

Quando você repete o comando após o início do trabalho de exportação, a resposta é mais ou menos esta:

```
{
  "jobId": "c86258f7-a9c9-4f8c-8f4c-bbfe76d51c8f",
  "status": "running",
  "logs": "https://us-east-1.console.aws.amazon.com/cloudwatch/home?..."
}
```

Se você abrir os logs no CloudWatch Logs usando o URI fornecido pela chamada de status, poderá monitorar detalhadamente o andamento da exportação:



Cancelar um trabalho de exportação em execução

Como cancelar o trabalho de exportação usando o AWS Management Console

1. Abra o console AWS Batch em https://console.aws.amazon.com/batch/.
2. Escolha Trabalhos.
3. Localize o trabalho em execução que você deseja cancelar, com base no `jobID`.
4. Selecione Cancelar trabalho.

Para cancelar um trabalho de exportação em execução usando a API de exportação do Neptune:

Envie uma solicitação `HTTP DELETE` ao `NeptuneExportApiUri` com o `jobID` anexado, desta forma:

```
curl -X DELETE \
```

```
  (your NeptuneExportApiUri) (the job ID)
```

Usar a ferramenta de linha de comando **`neptune-export`** para exportar dados do Neptune

É possível usar as seguintes etapas para exportar dados do cluster de banco de dados do Neptune para o Amazon S3 usando o utilitário de linha de comando `neptune-export`:

Pré-requisitos para usar o utilitário de linha de comando **`neptune-export`**

Antes de começar

- Tenha a versão 8 do JDK: você precisa instalar a versão 8 do Java SE Development Kit (JDK).
- Baixe o utilitário neptune-export: baixe e instale o arquivo neptune-export.jar.
- Garanta que o **`neptune-export`** tenha acesso à VPC do Neptune: execute o neptune-export de um local onde ele possa acessar a VPC em que o cluster de banco de dados do Neptune está localizado.

  Por exemplo, é possível executá-lo em uma instância do Amazon EC2 dentro da VPC do Neptune, ou em uma VPC separada que seja emparelhada com a VPC do Neptune, ou em um bastion host separado.
- Garanta que os grupos de segurança da VPC concedam acesso a **`neptune-export`**: confira se os grupos de segurança da VPC anexados à VPC do Neptune permitem acesso ao cluster de banco de dados a partir do endereço IP ou do grupo de segurança associado ao ambiente `neptune-export`.
- Configure as permissões necessárias do IAM: se o banco de dados tiver a autenticação de banco de dados do AWS Identity and Access Management (IAM) habilitada, garanta que o perfil sob o qual o `neptune-export` é executado esteja associado a uma política do IAM que viabilize conexões com o Neptune. Para obter informações sobre políticas do Neptune, consulte Como usar políticas do IAM.

  Se você quiser usar o parâmetro `clusterId` export nas solicitações de consulta, o perfil sob o qual o `neptune-export` é executado exige as seguintes permissões do IAM:

  - `rds:DescribeDBClusters`
  - `rds:DescribeDBInstances`
  - `rds:ListTagsForResource`

Se você quiser exportar de um cluster clonado, o perfil sob o qual `neptune-export` é executado exige as seguintes permissões do IAM:

- `rds:AddTagsToResource`
- `rds:DescribeDBClusters`
- `rds:DescribeDBInstances`
- `rds:ListTagsForResource`
- `rds:DescribeDBClusterParameters`
- `rds:DescribeDBParameters`
- `rds:ModifyDBParameterGroup`
- `rds:ModifyDBClusterParameterGroup`
- `rds:RestoreDBClusterToPointInTime`
- `rds:DeleteDBInstance`
- `rds:DeleteDBClusterParameterGroup`
- `rds:DeleteDBParameterGroup`
- `rds:DeleteDBCluster`
- `rds:CreateDBInstance`
- `rds:CreateDBClusterParameterGroup`
- `rds:CreateDBParameterGroup`

Para publicar os dados exportados no Amazon S3, o perfil sob o qual o `neptune-export` é executado exige as seguintes permissões do IAM para os locais do Amazon S3:

- `s3:PutObject`
- `s3:PutObjectTagging`
- `s3:GetObject`

- Defina a variável de ambiente **`SERVICE_REGION`**: defina a variável de ambiente `SERVICE_REGION` para identificar a região onde o cluster de banco de dados está localizado (consulte Connecting to Neptune para obter uma lista de identificadores de região).

Executar o utilitário **neptune-export** para iniciar uma operação de exportação

Use o seguinte comando para executar o neptune-export na linha de comando e iniciar uma operação de exportação:

```
java -jar neptune-export.jar nesvc \
  --root-path (path to a local directory) \
  --json (the JSON file that defines the export)
```

O comando tem dois parâmetros:

Parâmetros para neptune-export ao iniciar uma exportação

- **--root-path**: caminho para um diretório no qual os arquivos de exportação são gravados antes de serem publicados no Amazon S3.
- **--json**: um objeto JSON que define a exportação.

Exemplos de comando usando o utilitário de linha de comando **neptune-export**

Como exportar dados do grafo de propriedades diretamente do cluster de banco de dados de origem:

```
java -jar neptune-export.jar nesvc \
  --root-path /home/ec2-user/neptune-export \
  --json '{
            "command": "export-pg",
            "outputS3Path" : "s3://(your Amazon S3 bucket)/neptune-export",
            "params": {
              "endpoint" : "(your neptune DB cluster endpoint)"
            }
          }'
```

Como exportar dados do RDF diretamente do cluster de banco de dados de origem:

```
java -jar neptune-export.jar nesvc \
  --root-path /home/ec2-user/neptune-export \
```

```
  --json '{
            "command": "export-rdf",
            "outputS3Path" : "s3://(your Amazon S3 bucket)/neptune-export",
            "params": {
              "endpoint" : "(your neptune DB cluster endpoint)"
            }
          }'
```

Se você omitir o parâmetro de solicitação `command`, por padrão, o utilitário `neptune-export` exportará dados do grafo de propriedades do Neptune.

Como exportar de um clone do cluster de banco de dados:

```
java -jar neptune-export.jar nesvc \
  --root-path /home/ec2-user/neptune-export \
  --json '{
            "command": "export-pg",
            "outputS3Path" : "s3://(your Amazon S3 bucket)/neptune-export",
            "params": {
              "endpoint" : "(your neptune DB cluster endpoint)",
              "cloneCluster" : true
            }
          }'
```

Como exportar do cluster de banco de dados usando a autenticação do IAM:

```
java -jar neptune-export.jar nesvc \
  --root-path /home/ec2-user/neptune-export \
  --json '{
            "command": "export-pg",
            "outputS3Path" : "s3://(your Amazon S3 bucket)/neptune-export",
            "params": {
              "endpoint" : "(your neptune DB cluster endpoint)"
              "useIamAuth" : true
            }
          }'
```

Arquivos exportados pelo Neptune-Export e **`neptune-export`**

Quando uma exportação é concluída, os arquivos de exportação são publicados no local do Amazon S3 que você especificou. Todos os arquivos publicados no Amazon S3 são criptografados com criptografia do lado do servidor do Amazon S3 (`SSE-S3`). As pastas e os arquivos publicados no Amazon S3 variam dependendo se você está exportando grafos de propriedades ou dados do RDF. Se você abrir o local do Amazon S3 onde os arquivos são publicados, você verá o seguinte conteúdo:

Localizações dos arquivos exportados no Amazon S3

- **`nodes/`**: essa pasta contém arquivos de dados de nós em um formato CSV (valores separados por vírgula) ou JSON.

  No Neptune, os nós podem ter um ou mais rótulos. Os nós com rótulos individuais diferentes (ou combinações diferentes de vários rótulos) são gravados em arquivos distintos, o que significa que nenhum arquivo individual contém dados para nós com combinações diferentes de rótulos. Se um nó tiver vários rótulos, estes serão classificados em ordem alfabética antes de serem atribuídos a um arquivo.
- **`edges/`**: essa pasta contém arquivos de dados de bordas em um formato CSV (valores separados por vírgula) ou JSON.

  Assim como nos arquivos de nós, os dados de bordas são gravados em arquivos diferentes com base em uma combinação dos respectivos rótulos. Para fins de treinamento do modelo, os dados de bordas são atribuídos a arquivos diferentes com base em uma combinação do rótulo da borda com os rótulos dos nós inicial e final da borda.
- **`statements/`**: essa pasta contém arquivos de dados do RDF no formato Turtle, N-Quads, N-Triples ou JSON.
- **`config.json`**: esse arquivo contém o esquema do grafo conforme inferido pelo processo de exportação.
- **`lastEventId.json`**: esse arquivo contém o `commitNum` e o `opNum` do último evento nos fluxos do Neptune do banco de dados. O processo de exportação só incluirá esse arquivo se você definir o parâmetro de exportação `includeLastEventId` como `true` e o banco de dados do qual você está exportando dados tiver fluxos do Neptune habilitados.

Parâmetros usados para controlar o processo de exportação do Neptune

Se você estiver usando o serviço Neptune-Export ou o utilitário de linha de comando `neptune-export`, os parâmetros usados para controlar a exportação serão basicamente os mesmos. Eles contêm um objeto JSON transmitido ao endpoint Neptune-Export ou ao `neptune-export` na linha de comando.

O objeto transmitido ao processo de exportação tem até cinco campos de nível superior:

```
-d '{
      "command" : "(either export-pg or export-rdf)",
      "outputS3Path" : "s3:/(your Amazon S3 bucket)/(path to the folder for exported
 data)",
      "jobsize" : "(for Neptune-Export service only)",
      "params" : { (a JSON object that contains export-process parameters) },
      "additionalParams": { (a JSON object that contains parameters for training
 configuration) }
   }'
```

Sumário

O parâmetro **`command`**

O parâmetro `command` de nível superior determina se os dados do grafo de propriedades ou os dados do RDF devem ser exportados. Se você omitir o parâmetro `command`, o processo de exportação assumirá como padrão a exportação de dados do grafo de propriedades.

- **`export-pg`**: exportar dados do grafo de propriedades.
- **`export-rdf`**: exportar dados do RDF.

O parâmetro **`outputS3Path`**

O parâmetro `outputS3Path` de nível superior é obrigatório e deve conter o URI de um local do Amazon S3 no qual os arquivos exportados possam ser publicados:

```
  "outputS3Path" : "s3://(your Amazon S3 bucket)/(path to output folder)"
```

O valor deve começar com `s3://`, seguido por um nome de bucket válido e, opcionalmente, um caminho de pasta dentro do bucket.

O parâmetro **`jobSize`**

O parâmetro `jobSize` de nível superior é usado somente com o serviço Neptune-Export, não com o utilitário de linha de comando `neptune-export`, e é opcional. Ele permite caracterizar o tamanho do trabalho de exportação que você está iniciando, o que ajuda a determinar a quantidade de recursos computacionais dedicados ao trabalho e o nível máximo de simultaneidade.

```
  "jobsize" : "(one of four size descriptors)"
```

Os quatro descritores de tamanho válidos são:

- `small`: máximo de simultaneidade: oito. Adequado para volumes de armazenamento de até 10 GB.
- `medium`: máximo de simultaneidade: 32. Adequado para volumes de armazenamento de até 100 GB.
- `large`: máximo de simultaneidade: 64. Adequado para volumes de armazenamento acima de 100 GB, mas menores que 1 TB.
- `xlarge`: máximo de simultaneidade: 96. Adequado para volumes de armazenamento acima de 1 TB.

Por padrão, uma exportação iniciada no serviço Neptune-Export é executada como um trabalho `small`.

O desempenho de uma exportação depende não apenas da configuração `jobSize`, mas também do número de instâncias de banco de dados das quais você está exportando, do tamanho de cada instância e do nível efetivo de simultaneidade do trabalho.

Para exportações de grafos de propriedades, é possível configurar o número de instâncias do banco de dados usando o parâmetro cloneClusterReplicaContagem e configurar o nível efetivo de simultaneidade do trabalho usando o parâmetro concurrency.

O objeto **`params`**.

O parâmetro `params` de nível superior é um objeto JSON que contém parâmetros usados para controlar o próprio processo de exportação, conforme explicado em Exportar campos de parâmetros no objeto JSON `params` de nível superior. Alguns dos campos no objeto `params` são específicos de exportações de grafos de propriedades, outros para RDF.

O objeto **`additionalParams`**.

O parâmetro `additionalParams` de nível superior é um objeto JSON com parâmetros que você pode usar para controlar ações aplicadas aos dados após a exportação. No momento, `additionalParams` é usado apenas para exportar dados de treinamento para o Neptune ML.

Exportar campos de parâmetros no objeto JSON **`params`** de nível superior

O objeto JSON `params` de exportação do Neptune permite controlar a exportação, incluindo o tipo e o formato dos dados exportados.

Lista de campos possíveis no objeto **`params`** de parâmetros de exportação

Veja listados abaixo todos os possíveis campos de nível superior que podem aparecer em um objeto `params`. Somente um subconjunto desses campos aparece em qualquer objeto.

Lista de campos comuns a todos os tipos de exportação

- `cloneCluster`
- `cloneClusterInstanceType`
- `cloneClusterReplicaCount`
- `clusterId`
- `endpoint`
- `endpoints`
- `profile`
- `useIamAuth`
- `includeLastEventId`

Lista de campos para exportação de grafos de propriedades

- `concurrency`
- `edgeLabels`
- `filter`
- `filterConfigFile`
- `gremlinFilter`
- `gremlinNodeFilter`
- `gremlinEdgeFilter`
- `format`
- `nodeLabels`

- `scope`

Lista de campos para exportações do RDF

- `format`
- `rdfExportScope`
- `sparql`
- `namedGraph`

Campos comuns a todos os tipos de exportação

Campo **`cloneCluster`** em **`params`**

(Opcional). Padrão: `false`.

Se o parâmetro `cloneCluster` estiver definido como `true`, o processo de exportação usará um clone rápido do cluster de banco de dados:

```
  "cloneCluster" : true
```

Por padrão, o processo de exportação exporta dados do cluster de banco de dados especificado usando parâmetros `endpoint`, `endpoints` ou `clusterId`. No entanto, se o cluster de banco de dados estiver em uso durante a exportação e os dados estiverem mudando, o processo de exportação não poderá garantir a consistência dos dados que estão sendo exportados.

Para garantir que os dados exportados sejam consistentes, use o parâmetro `cloneCluster` para exportar de um clone estático do cluster de banco de dados.

O cluster de banco de dados clonado é criado na mesma VPC do cluster de banco de dados de origem e herda as configurações de autenticação do grupo de segurança, do grupo de sub-redes e do banco de dados do IAM da origem. Quando a exportação estiver concluída, o Neptune excluirá o cluster de banco de dados clonado.

Por padrão, um cluster de banco de dados clonado consiste em uma única instância do mesmo tipo de instância da instância principal no cluster de banco de dados de origem. É possível alterar o tipo de instância usado para o cluster de banco de dados clonado especificando um diferente com `cloneClusterInstanceType`.

> **Note**
>
> Se você não usa a opção `cloneCluster` e está exportando diretamente do cluster de banco de dados principal, talvez seja necessário aumentar o tempo limite nas instâncias das quais os dados estão sendo exportados. Para grandes conjuntos de dados, o tempo limite deve ser definido como algumas horas.

Campo **`cloneClusterInstanceType`** em **`params`**

(Opcional).

Se o parâmetro `cloneCluster` estiver presente e definido como `true`, você poderá usar o parâmetro `cloneClusterInstanceType` para especificar o tipo de instância usado para o cluster de banco de dados clonado:

Por padrão, um cluster de banco de dados clonado consiste em uma única instância do mesmo tipo de instância da instância principal no cluster de banco de dados de origem.

```
"cloneClusterInstanceType" : "(for example, r5.12xlarge)"
```

Campo **`cloneClusterReplicaCount`** em **`params`**

(Opcional).

Se o parâmetro `cloneCluster` estiver presente e definido como `true`, você poderá usar o parâmetro `cloneClusterReplicaCount` para especificar o número de instâncias de réplica de leitura criadas no cluster de banco de dados clonado:

```
"cloneClusterReplicaCount" : (for example, 3)
```

Por padrão, um cluster de banco de dados clonado consiste em uma única instância principal. O parâmetro `cloneClusterReplicaCount` permite especificar quantas instâncias adicionais de réplica de leitura devem ser criadas.

Campo **`clusterId`** em **`params`**

(Opcional).

O parâmetro `clusterId` especifica o ID de um cluster de banco de dados a ser usado:

```
"clusterId" : "(the ID of your DB cluster)"
```

Se você usar o parâmetro `clusterId`, o processo de exportação usará todas as instâncias disponíveis nesse cluster de banco de dados para extrair dados.

> **Note**
>
> Os parâmetros `endpoint`, `endpoints` e `clusterId` são mutuamente exclusivos. Use um e somente um deles.

Campo **`endpoint`** em **`params`**

(Opcional).

Use `endpoint` para especificar um endpoint de uma instância do Neptune no cluster de banco de dados que o processo de exportação pode consultar para extrair dados (consulte Conexões do endpoint). Esse é somente o nome do DNS e não inclui o protocolo nem a porta:

```
"endpoint" : "(a DNS endpoint of your DB cluster)"
```

Use um endpoint de cluster ou de instância, mas não o endpoint de leitor principal.

> **Note**
>
> Os parâmetros `endpoint`, `endpoints` e `clusterId` são mutuamente exclusivos. Use um e somente um deles.

Campo **`endpoints`** em **`params`**

(Opcional).

Use `endpoints` para especificar uma matriz JSON de endpoints no cluster de banco de dados que o processo de exportação pode consultar para extrair dados (consulte Conexões do endpoint). Esses são somente nomes do DNS e não incluem o protocolo nem a porta:

```
"endpoints": [
  "(one endpoint in your DB cluster)",
```

```
    "(another endpoint in your DB cluster)",
    "(a third endpoint in your DB cluster)"
    ]
```

Se você tiver várias instâncias no cluster (uma principal e uma ou mais réplicas de leitura), poderá melhorar o desempenho de exportação usando o parâmetro `endpoints` para distribuir consultas em uma lista desses endpoints.

> **Note**
>
> Os parâmetros `endpoint`, `endpoints` e `clusterId` são mutuamente exclusivos. Use um e somente um deles.

Campo **`profile`** em **`params`**

(Obrigatório para exportar dados de treinamento para o Neptune ML, a menos que o campo *`neptune_ml`* esteja presente no campo *`additionalParams`*).

O parâmetro `profile` fornece conjuntos de parâmetros pré-configurados para workloads específicas. No momento, o processo de exportação é compatível apenas com o perfil `neptune_ml`.

Se você estiver exportando dados de treinamento para o Neptune ML, adicione o seguinte parâmetro ao objeto `params`:

```
"profile" : "neptune_ml"
```

Campo **`useIamAuth`** em **`params`**

(Opcional). Padrão: `false`.

Se o banco de dados do qual você está exportando dados tiver a autenticação do IAM habilitada, deverá incluir o parâmetro `useIamAuth` definido como `true`:

```
"useIamAuth" : true
```

Campo **`includeLastEventId`** em **`params`**

Se você definir `includeLastEventId` como verdadeiro e o banco de dados do qual está exportando dados tiver os Fluxos do Neptune habilitados, o processo de exportação gravará um

arquivo `lastEventId.json` no local de exportação especificado. Esse arquivo contém `commitNum` e opNum do último evento do fluxo.

```
  "includeLastEventId" : true
```

Um banco de dados clonado criado pelo processo de exportação herda a configuração de fluxos do pai. Se o pai tiver fluxos habilitados, o clone também terá fluxos habilitados. O conteúdo do fluxo no clone refletirá o conteúdo do pai (incluindo os mesmos IDs de evento) no momento em que o clone foi criado.

Campos para exportação de grafos de propriedades

Campo **`concurrency`** em **`params`**

(Opcional). Padrão: 4.

O parâmetro `concurrency` especifica o número de consultas paralelas que o processo de exportação deve usar:

```
  "concurrency" : (for example, 24)
```

Uma boa diretriz é definir o nível de simultaneidade como o dobro do número de vCPUs em todas as instâncias das quais você está exportando dados. Uma instância r5.xlarge, por exemplo, tem quatro vCPUs. Se você estiver exportando de um cluster de três instâncias r5.xlarge, poderá definir o nível de simultaneidade como 24 (= 3 x 2 x 4).

Se você estiver usando o serviço Neptune-Export, o nível de simultaneidade será limitado pela configuração jobSize. Um trabalho pequeno, por exemplo, é compatível com um nível de simultaneidade de oito. Se você tentar especificar um nível de simultaneidade de 24 para um trabalho pequeno usando o parâmetro `concurrency`, o nível efetivo permanecerá em oito.

Se você exportar de um cluster clonado, o processo de exportação calculará um nível de simultaneidade apropriado com base no tamanho das instâncias clonadas e no tamanho do trabalho.

Campo **`edgeLabels`** em **`params`**

(Opcional).

Use `edgeLabels` para exportar somente as bordas que têm rótulos especificados:

```
  "edgeLabels" : ["(a label)", "(another label"]
```

Cada rótulo na matriz JSON deve ser um rótulo único e simples.

O parâmetro `scope` tem precedência sobre o parâmetro `edgeLabels`, portanto, se o valor `scope` não incluir bordas, o parâmetro `edgeLabels` não terá efeito.

Campo **`filter`** em **`params`**

(Opcional).

Use `filter` para especificar que somente nós e/ou bordas com rótulos específicos devem ser exportados e para filtrar as propriedades exportadas para cada nó ou borda.

A estrutura geral de um objeto `filter`, em linha ou em um arquivo de configuração de filtro, é a seguinte:

```
  "filter" : {
    "nodes": [ (array of node label and properties objects) ],
    "edges": [ (array of edge definition an properties objects) ]
  }
```

- **`nodes`**: contém uma matriz JSON de nós e propriedades de nós no seguinte formato:

  ```
      "nodes : [
        {
          "label": "(node label)",
          "properties": [ "(a property name)", "(another property name)", ( ... ) ]
        }
      ]
  ```

  - `label`: o rótulo ou rótulos do grafo de propriedades do nó.

    Assume um único valor ou, se o nó tiver vários rótulos, uma matriz de valores.
  - `properties`: contém uma matriz dos nomes das propriedades do nó que você deseja exportar.
- **`edges`**: contém uma matriz JSON de definições de borda no seguinte formato:

  ```
      "edges" : [
        {
          "label": "(edge label)",
  ```

```
        "properties": [ "(a property name)", "(another property name)", ( ... ) ]
      }
    ]
```

- `label`: o rótulo do grafo de propriedades da borda. Assume um único valor.
- `properties`: contém uma matriz dos nomes das propriedades da borda que você deseja exportar.

Campo **`filterConfigFile`** em **`params`**

(Opcional).

Use `filterConfigFile` para especificar um arquivo JSON que contém uma configuração de filtro no mesmo formato assumido pelo parâmetro `filter`:

```
 "filterConfigFile" : "s3://(your Amazon S3 bucket)/neptune-export/(the name of the
JSON file)"
```

Consulte filtrar para conhecer o formato do arquivo `filterConfigFile`.

Campo **`format`** usado para dados do grafo de propriedades em **`params`**

(Opcional). Padrão: `csv` (valores separados por vírgula)

O parâmetro `format` especifica o formato de saída dos dados do grafo de propriedades exportado:

```
 "format" : (one of: csv, csvNoHeaders, json, neptuneStreamsJson)
```

- **`csv`**: saída formatada com valores separados por vírgula (CSV), com cabeçalhos de coluna formatados de acordo com o formato de dados de carga do Gremlin.
- **`csvNoHeaders`**: dados formatados em CSV sem cabeçalhos de coluna.
- **`json`**: dados formatados em JSON.
- **`neptuneStreamsJson`**: dados formatados em JSON que usam o formato de serialização de alterações GREMLIN_JSON.

Campo **`gremlinFilter`** em **`params`**

(Opcional).

O parâmetro `gremlinFilter` permite fornecer um trecho do Gremlin, como uma etapa `has()`, que é usado para filtrar nós e bordas:

```
"gremlinFilter" : (a Gremlin snippet)
```

Os nomes dos campos e os valores das strings devem estar entre aspas duplas de escape. Para datas e horários, você pode usar o método datetime.

O exemplo a seguir exporta somente os nós e as bordas com uma propriedade de data de criação cujo valor é maior que 10/10/2021:

```
"gremlinFilter" : "has(\"created\", gt(datetime(\"2021-10-10\")))"
```

Campo **`gremlinNodeFilter`** em **`params`**

(Opcional).

O parâmetro `gremlinNodeFilter` permite fornecer um trecho do Gremlin, como uma etapa `has()`, que é usado para filtrar nós:

```
"gremlinNodeFilter" : (a Gremlin snippet)
```

Os nomes dos campos e os valores das strings devem estar entre aspas duplas de escape. Para datas e horários, você pode usar o método datetime.

O exemplo a seguir exporta somente aqueles nós com uma propriedade booliana `deleted` cujo valor é `true`:

```
"gremlinNodeFilter" : "has(\"deleted\", true)"
```

Campo **`gremlinEdgeFilter`** em **`params`**

(Opcional).

O parâmetro `gremlinEdgeFilter` permite fornecer um trecho do Gremlin, como uma etapa `has()`, que é usado para filtrar bordas:

```
"gremlinEdgeFilter" : (a Gremlin snippet)
```

Os nomes dos campos e os valores das strings devem estar entre aspas duplas de escape. Para datas e horários, você pode usar o método datetime.

O exemplo a seguir exporta somente aquelas bordas com uma propriedade numérica cujo valor cinco:

```
"gremlinEdgeFilter" : "has(\"strength\", 5)"
```

Campo **`nodeLabels`** em **`params`**

(Opcional).

Use `nodeLabels` para exportar somente aqueles nós que têm rótulos especificados:

```
"nodeLabels" : ["(a label)", "(another label"]
```

Cada rótulo na matriz JSON deve ser um rótulo único e simples.

O parâmetro `scope` tem precedência sobre o parâmetro `nodeLabels`, portanto, se o valor `scope` não incluir nós, o parâmetro `nodeLabels` não terá efeito.

Campo **`scope`** em **`params`**

(Opcional). Padrão: `all`.

O parâmetro `scope` especifica se você deve exportar somente nós, somente bordas ou nós e bordas:

```
"scope" : (one of: nodes, edges, or all)
```

- `nodes`: exportar somente nós e as respectivas propriedades.
- `edges`: exportar somente bordas e as respectivas propriedades.
- `all`: exportar nós e bordas e as respectivas propriedades (o padrão).

Campos para exportação do RDF

Campo **`format`** usado para dados do RDF em **`params`**

(Opcional). Padrão: `turtle`

O parâmetro `format` especifica o formato de saída dos dados do RDF exportados:

```
"format" : (one of: turtle, nquads, ntriples, neptuneStreamsJson)
```

- **`turtle`**: saída formatada em Turtle.
- **`nquads`**: dados formatados em N-Quads sem cabeçalhos de coluna.
- **`ntriples`**: dados formatados em N-Triples.
- **`neptuneStreamsJson`**: dados formatados em JSON que usam o formato de serialização de alterações SPARQL NQUADS.

Campo **`rdfExportScope`** em **`params`**

(Opcional). Padrão: `graph`.

O parâmetro `rdfExportScope` especifica o escopo da exportação do RDF:

```
"rdfExportScope" : (one of: graph, edges, or query)
```

- `graph`: exportar todos os dados do RDF.
- `edges`: exportar somente os triplos que representam bordas.
- `query`: exportar dados recuperados por uma consulta do SPARQL fornecida com o campo `sparql`.

Campo **`sparql`** em **`params`**

(Opcional).

O parâmetro `sparql` permite especificar uma consulta do SPARQL para recuperar os dados a serem exportados:

```
"sparql" : (a SPARQL query)
```

Se você fornecer uma consulta com o campo `sparql`, também deverá definir o campo `rdfExportScope` como `query`.

Campo **`namedGraph`** em **`params`**

(Opcional).

O `namedGraph` parâmetro permite que você especifique um IRI para limitar a exportação a um único gráfico nomeado:

```
  "namedGraph" : (Named graph IRI)
```

O `namedGraph` parâmetro só pode ser usado com o `rdfExportScope` campo definido como`graph`.

Exemplos de filtragem de dados exportados

Veja alguns exemplos que ilustram formas de filtrar os dados exportados.

Filtrar a exportação de dados do grafo de propriedades

Exemplo de uso de **`scope`** para exportar somente bordas

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "scope": "edges"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Exemplo de uso de **`nodeLabels`** e **`edgeLabels`** para exportar somente nós e bordas com rótulos específicos

O parâmetro `nodeLabels` no exemplo a seguir especifica que somente nós com um rótulo `Person` ou `Post` devem ser exportados. O parâmetro `edgeLabels` especifica que somente bordas com o rótulo `likes` devem ser exportadas:

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "nodeLabels": ["Person", "Post"],
    "edgeLabels": ["likes"]
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Exemplo de uso de **`filter`** para exportar somente nós, bordas e propriedades especificados

O objeto `filter` neste exemplo exporta nós `country` com as respectivas propriedades `type`, `code` e `desc` e também bordas `route` com as respectivas propriedades `dist`.

```
{
  "command": "export-pg",
  "params": {
```

```
    "endpoint": "(your Neptune endpoint DNS name)",
    "filter": {
      "nodes": [
        {
          "label": "country",
          "properties": [
            "type",
            "code",
            "desc"
          ]
        }
      ],
      "edges": [
        {
          "label": "route",
          "properties": [
            "dist"
          ]
        }
      ]
    }
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Exemplo que usa **`gremlinFilter`**

Este exemplo usa `gremlinFilter` para exportar somente os nós e as bordas criados após 10/10/2021 (ou seja, com uma propriedade `created` cujo valor é maior que 10/10/2021):

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "gremlinFilter" : "has(\"created\", gt(datetime(\"2021-10-10\")))"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Exemplo que usa **`gremlinNodeFilter`**

Este exemplo usa `gremlinNodeFilter` para exportar somente nós excluídos (nós com uma propriedade `deleted` booliana cujo valor é `true`):

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "gremlinNodeFilter" : "has(\"deleted\", true)"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Exemplo que usa **`gremlinEdgeFilter`**

O seguinte exemplo usa `gremlinEdgeFilter` para exportar somente as bordas com uma propriedade numérica `strength` cujo valor é cinco:

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "gremlinEdgeFilter" : "has(\"strength\", 5)"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Combinando **`filter`**, **`gremlinNodeFilter`**, **`nodeLabels`**, **`edgeLabels`** e **`scope`**

O objeto `filter` neste exemplo exporta:

- Nós `country` com as respectivas propriedades `type`, `code` e `desc`
- Nós `airport` com as respectivas propriedades `code`, `icao` e `runways`
- Bordas `route` com a respectiva propriedade `dist`

O parâmetro `gremlinNodeFilter` filtra os nós para que somente os nós com uma propriedade `code` cujo valor comece com A sejam exportados.

Os parâmetros `nodeLabels` e `edgeLabels` restringem ainda mais a saída para que somente nós `airport` e bordas `route` sejam exportados.

Por fim, o parâmetro `scope` elimina as bordas da exportação, o que deixa somente os nós `airport` designados na saída.

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "filter": {
      "nodes": [
        {
          "label": "airport",
          "properties": [
            "code",
            "icao",
            "runways"
          ]
        },
        {
          "label": "country",
          "properties": [
            "type",
            "code",
            "desc"
          ]
        }
      ],
      "edges": [
        {
          "label": "route",
          "properties": [
            "dist"
          ]
        }
      ]
    },
    "gremlinNodeFilter": "has(\"code\", startingWith(\"A\"))",
    "nodeLabels": [
      "airport"
    ],
    "edgeLabels": [
      "route"
    ],
    "scope": "nodes"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Filtrar a exportação de dados do RDF

Usar **`rdfExportScope`** e **`sparql`** para exportar bordas específicas

Este exemplo exporta triplos cujo predicado é <http://kelvinlawrence.net/air-routes/objectProperty/route> e cujo objeto não é um literal:

```
{
  "command": "export-rdf",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "rdfExportScope": "query",
    "sparql": "CONSTRUCT { ?s <http://kelvinlawrence.net/air-routes/objectProperty/
route> ?o } WHERE { ?s ?p ?o . FILTER(!isLiteral(?o)) }"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Usando **`namedGraph`** para exportar um único gráfico nomeado

Este exemplo exporta triplos pertencentes ao gráfico nomeado < http://aws.amazon.com/neptune/vocab/v01/ DefaultNamedGraph >:

```
{
  "command": "export-rdf",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "rdfExportScope": "graph",
    "namedGraph": "http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Solução de problemas do processo de exportação do Neptune

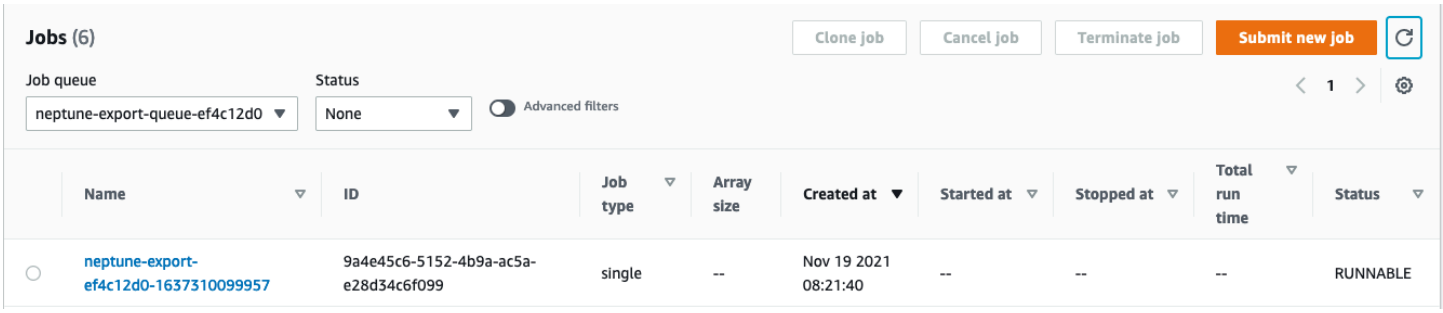
O processo de exportação do Amazon Neptune usa AWS Batch para provisionar os recursos computacionais e de armazenamento necessários para exportar os dados do Neptune. Durante a execução de uma exportação, é possível usar o link no campo `logs` para acessar os logs do CloudWatch para o trabalho de exportação.

No entanto, os logs do CloudWatch para o trabalho AWS Batch que executa a exportação só estão disponíveis quando o trabalho AWS Batch está em execução. Se a exportação do Neptune informar que uma exportação está em um estado pendente, não haverá um link de logs por meio do qual acessar os logs do CloudWatch. Se um trabalho de exportação permanecer no estado `pending` por mais de alguns minutos, poderá haver um problema no provisionamento dos recursos AWS Batch subjacentes.

Quando o trabalho de exportação sair do estado pendente, você poderá conferir o status da seguinte forma:

Como conferir o status de um trabalho AWS Batch

1. Abra o console do AWS Batch em https://console.aws.amazon.com/batch/.
2. Selecione a fila de trabalhos do neptune-export.
3. Procure o trabalho cujo nome corresponda ao `jobName` gerado pela exportação do Neptune quando você iniciou a exportação.

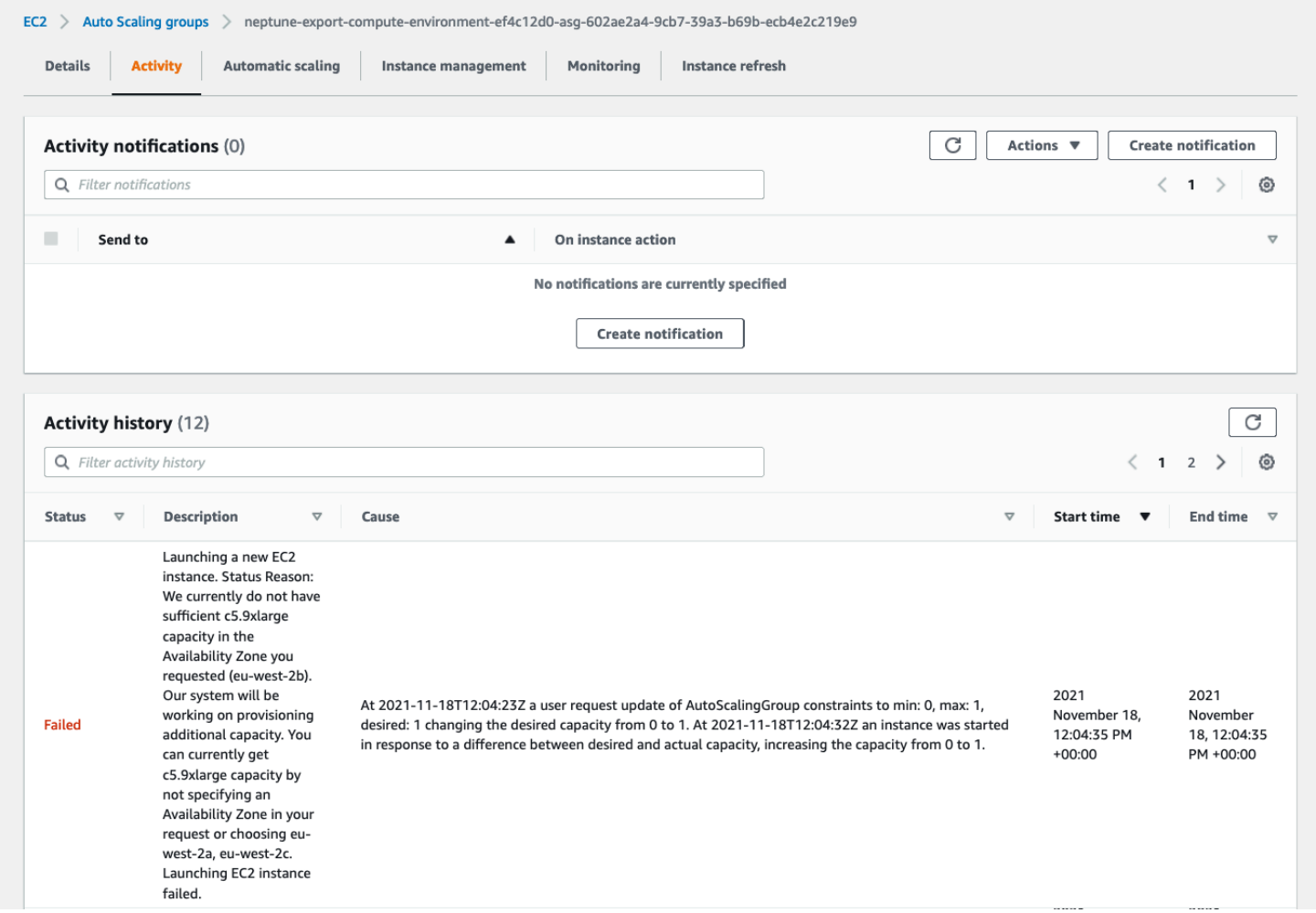


Se o trabalho permanecer paralisado em um estado `RUNNABLE`, o motivo pode ser que problemas de rede ou segurança estão impedindo que a instância do contêiner se una ao cluster subjacente do Amazon Elastic Container Service (Amazon ECS). Consulte a seção sobre como conferir as configurações de rede e segurança do ambiente computacional neste artigo de suporte.

Outro fator que você pode conferir é se há problemas com o ajuste de escala automático:

Como conferir o grupo de ajuste de escala automático do Amazon EC2 para o ambiente computacional AWS Batch

1. Abra o console do Amazon EC2 em https://console.aws.amazon.com/ec2/.
2. Selecione o grupo do Auto Scaling para o ambiente computacional do neptune-export.
3. Abra a guia Atividade e confira se há eventos malsucedidos no histórico de atividades.



Erros comuns do Neptune Export

`org.eclipse.rdf4j.query.QueryEvaluationException: Tag mismatch!`

Se um trabalho `export-rdf` falhar regularmente com um `Tag mismatch! QueryEvaluationException`, a instância do Neptune está subdimensionada para as consultas grandes e de longa execução usadas pelo Neptune Export.

É possível evitar esse erro aumentando a escala verticalmente para uma instância maior do Neptune ou configurando o trabalho para exportar de um grande cluster clonado, desta forma:

```
'{
  "command": "export-rdf",
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "cloneCluster": True,
    "cloneClusterInstanceType" : "r5.24xlarge"
  }
}'
```

Gerenciar o banco de dados do Amazon Neptune

Esta seção mostra como gerenciar e manter o cluster de banco de dados do Neptune usando o AWS Management Console e a AWS CLI.

O Neptune opera em clusters de servidores de banco de dados conectados em uma topologia de replicação. Assim, gerenciar o Neptune normalmente envolve implantar alterações em vários servidores e garantir que todas as réplicas do Neptune estejam acompanhando o servidor primário.

Como o Neptune dimensiona o armazenamento subjacente de forma transparente à medida que os dados crescem, gerenciar o Neptune exige relativamente pouco gerenciamento de armazenamento em disco. Da mesma forma, como o Neptune realiza backups contínuos automaticamente, um cluster do Neptune não exige planejamento extensivo nem tempo de inatividade para realizar backups.

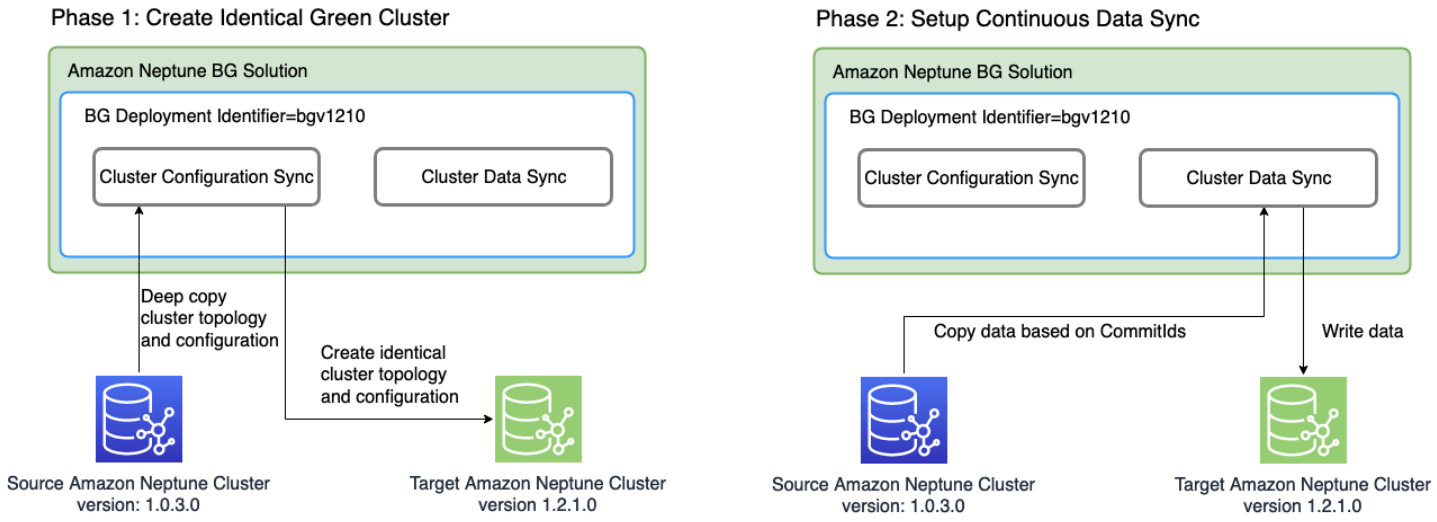
Tópicos

Usar a solução Neptune Blue/Green para realizar atualizações azuis/verdes

As atualizações do mecanismo do Amazon Neptune podem precisar de tempo de inatividade da aplicação porque o banco de dados não fica disponível enquanto as atualizações estão sendo instaladas e verificadas. Isso será fato se eles forem iniciados manualmente ou automaticamente.

O Neptune fornece uma solução de implantação azul/verde que você pode executar usando uma pilha AWS CloudFormation e que reduz consideravelmente esse tempo de inatividade. Ela cria um ambiente de preparação verde que é sincronizado com o ambiente de produção azul. Depois, é possível atualizar esse ambiente de preparação para realizar uma atualização secundária ou principal da versão do mecanismo, uma alteração no modelo de dados de grafos ou uma atualização do sistema operacional e testar o resultado. Por fim, é possível migrá-lo rapidamente para se tornar seu ambiente de produção, com muito pouco tempo de inatividade.

A solução Neptune Blue/Green passa por duas fases, conforme ilustrado neste diagrama:



A fase 1 cria um cluster de banco de dados verde idêntico ao cluster de produção

A solução cria um cluster de banco de dados com um identificador exclusivo de implantação azul/verde e com a mesma topologia do cluster de produção. Ou seja, ele tem o mesmo número e tamanho de instâncias de banco de dados, os mesmos grupos de parâmetros e todas as mesmas configurações do cluster de banco de dados de produção (azul), exceto que ele foi atualizado para a versão do mecanismo de destino que você especificou, que deve ser superior à versão atual do mecanismo (azul). É possível especificar uma versão secundária e principal do mecanismo para o destino. Se necessário, a solução executará todas as atualizações intermediárias necessárias para

alcançar a versão especificada do mecanismo de destino. Esse novo cluster se torna o ambiente de preparação verde.

A fase 2 configura a sincronização contínua de dados

Depois que o ambiente verde estiver totalmente preparado, a solução vai configurar a replicação contínua entre o cluster de origem (azul) e o cluster de destino (verde) usando fluxos do Neptune. Quando a diferença de replicação entre eles chegar a zero, o ambiente de teste estará pronto para testes. Nesse ponto, você deve pausar a gravação no cluster azul para evitar mais atrasos na replicação.

A versão do mecanismo de destino pode ter novos recursos ou dependências que afetam as aplicações. Confira a página de versão do mecanismo de destino e as páginas de versão do mecanismo de intervenção em Versões do mecanismo para ver o que mudou desde a versão atual do mecanismo. É melhor executar testes de integração ou verificar as aplicações manualmente no cluster verde antes de promovê-lo ao ambiente de produção.

Depois de testar e qualificar as alterações no cluster verde, basta mudar o endpoint do banco de dados nas aplicações do cluster azul para o verde.

Após a transição, a solução Neptune Blue/Green não exclui o antigo ambiente de produção azul. Você ainda terá acesso a ele para validação e testes adicionais, se necessário. As cobranças de faturamento padrão se aplicam às instâncias até a exclusão. A solução Blue/Green também usa outros serviços da AWS, cujos custos são cobrados a preços normais. Os detalhes sobre como excluir a solução quando terminar de usá-la são abordados na seção sobre limpeza.

Pré-requisitos para executar a pilha do Neptune Blue/Green

Antes de iniciar a pilha do Neptune Blue/Green:

- Habilite os fluxos do Neptune no cluster de produção (azul).
- Todas as instâncias no cluster azul devem estar no estado disponível. É possível conferir os estados das instâncias no console do Neptune ou usando a API describe-db-instances.
- Todas as instâncias também devem estar sincronizadas com o grupo de parâmetros do cluster de banco de dados.
- A solução Neptune Blue/Green precisa de um endpoint da VPC do DynamoDB na VPC em que o cluster azul está localizado. Consulte Usar endpoints da Amazon VPC para acessar o DynamoDB.
- Selecione um momento para executar a solução quando a workload de gravação no cluster de banco de dados azul de produção for a mais leve possível. Evite, por exemplo, executar a solução

durante um carregamento em massa ou quando houver probabilidade de um grande número de operações de gravação por qualquer outro motivo.

Usar um modelo AWS CloudFormation para executar a solução Neptune Blue/Green

É possível usar AWS CloudFormation para implantar a solução Neptune Blue/Green. O modelo do CloudFormation cria uma instância do Amazon EC2 na mesma VPC do banco de dados do Neptune de origem azul, instala a solução no local e a executa. É possível monitorar o andamento nos logs do CloudWatch, conforme explicado em Monitoring progress.

É possível usar esses links para avaliar o modelo de solução ou selecionar o botão Iniciar pilha para iniciá-lo no console do AWS CloudFormation:

Visão	Visualizar no Designer	

No console, escolha a região da AWS em que você deseja executar a solução no menu suspenso, no canto superior direito da janela.

Defina os parâmetros da pilha da seguinte forma:

- **`DeploymentID`**: um identificador exclusivo para cada implantação azul/verde do Neptune.

  Ele é usado como o identificador verde do cluster de banco de dados e como um prefixo para nomear novos recursos criados durante a implantação.
- **`NeptuneSourceClusterId`**: o identificador do cluster de banco de dados azul que você deseja atualizar.
- **`NeptuneTargetClusterVersion:`**: a versão do mecanismo do Neptune para a qual você deseja atualizar o cluster de banco de dados azul.

  Esse valor deve ser mais recente do que a versão do mecanismo atual do cluster de banco de dados azul.
- **`DeploymentMode`**: indica se essa é uma nova implantação ou uma tentativa de retomar uma implantação anterior. Quando você estiver usando o mesmo `DeploymentID` de uma implantação anterior, defina `DeploymentMode` como `resume`.

Os valores válidos são: `new` (o padrão) e `resume`.

- **`GraphQueryType`**: o tipo de dados de grafos para o banco de dados.

  Os valores válidos são: `propertygraph` (o padrão) e `rdf`.

- **`SubnetId`**: um ID de sub-rede da mesma VPC em que o cluster de banco de dados azul está localizado. (Consulte Connecting to a Neptune DB Cluster from an Amazon EC2 instance in the same VPC.)

  Forneça o ID de uma sub-rede pública se você quiser usar o SSH para a instância por meio do EC2 Connect.

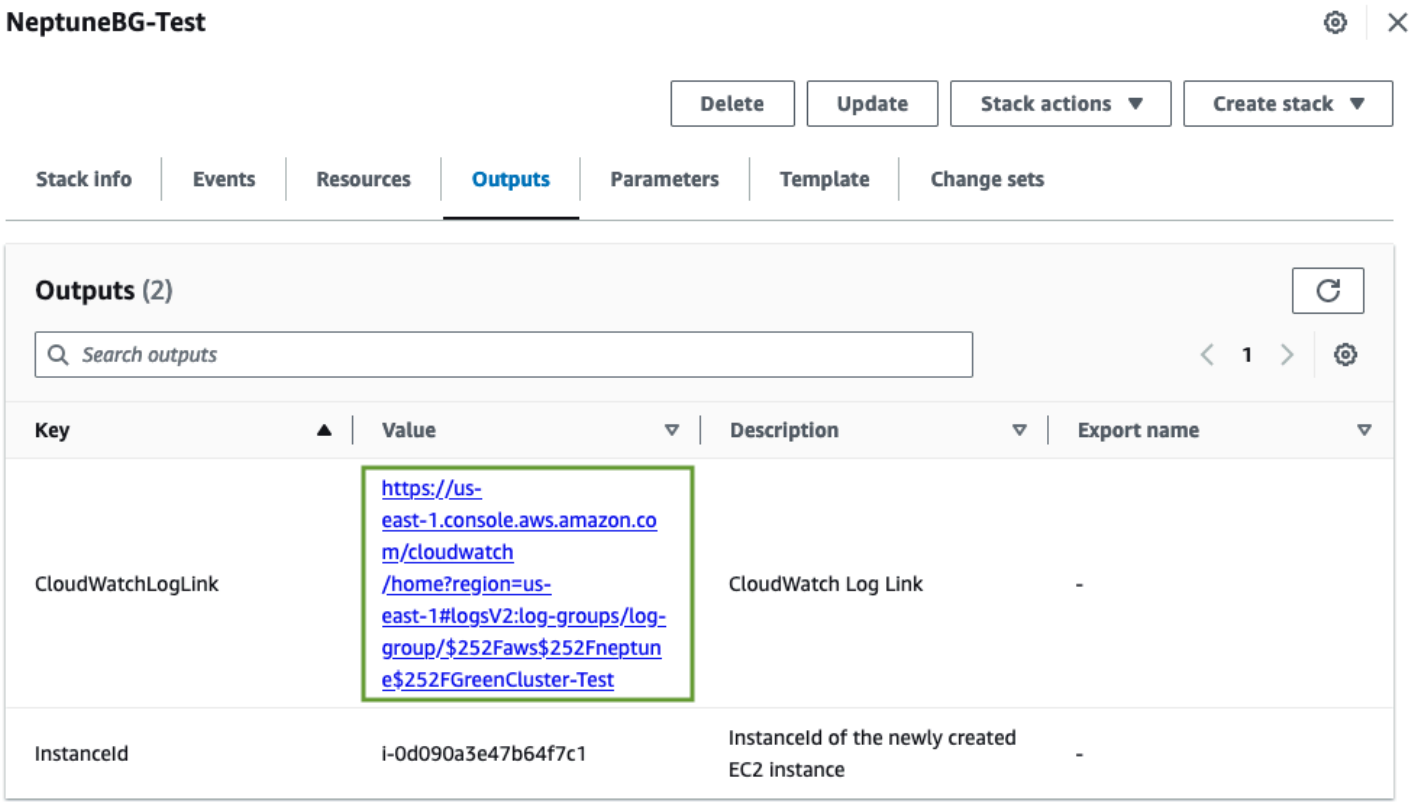
- **`InstanceSecurityGroup`**: um grupo de segurança para a instância do EC2.

  O grupo de segurança deve ter acesso ao cluster de banco de dados azul e você deve poder usar SSH para a instância. Consulte Criar um grupo de segurança usando o console da VPC.

Espere até que a pilha esteja completa. Assim que finalizar, a solução será iniciada. Depois, você pode monitorar o processo de implantação usando os logs do CloudWatch, conforme descrito na próxima seção.

Monitorar o andamento de uma implantação azul/verde do Neptune

É possível monitorar o andamento da solução Neptune Blue/Green acessando o console do CloudWatch e examinando os logs no grupo de logs `/aws/neptune/`*`(Neptune Blue/Green deployment ID)`* do CloudWatch. É possível encontrar um link para os logs do CloudWatch nas saídas da pilha AWS CloudFormation da solução:



Se você forneceu uma sub-rede pública como parâmetro de pilha, também poderá usar SSH para a instância do Amazon EC2 criada como parte da pilha e consultar o login em `/var/log/cloud-init-output.log`.

O log mostra as ações realizadas pela solução Neptune Blue/Green, conforme mostrado nesta captura de tela:

```
============================================================

Neptune Blue Green Deployment Solution Version: 0.1.06012023

============================================================

Checking whether cluster with id = bg-06-01-14-20-29test-bg1-bgInt already exists.

BlueGreen deployment_mode = new

Didn't find any cluster with id bg-06-01-14-20-29test-bg1-bgInt

Cloned_cluster_id: bg-06-01-14-20-29test-bg1-bgInt

Replication_stack_name: bg-06-01-14-20-29test-bg1-bgInt-replication

DescribeDbClusters response for test-bg1-bgIntegTest-06-01-14-20-29: {'AllocatedStorage': 1,
'AvailabilityZones': ['us-east-1b', 'us-east-1c', 'us-east-1f'], 'BackupRetentionPeriod': 1,
'DBClusterIdentifier': 'test-bg1-bgintegtest-06-01-14-20-29', 'DBClusterParameterGroup': 'green-     -blue-
green-deployment-test-123456789012345-pg-tes710', 'DBSubnetGroup': 'default', 'Status': 'available',
'EarliestRestorableTime': datetime.datetime(2023, 6, 1, 8, 51, 23, 394000, tzinfo=tzlocal()), 'Endpoint':
'test-bg1-bgintegtest-06-01-14-20-29.cluster-critvszpmydm.us-east-1.neptune.amazonaws.com', 'ReaderEndpoint':
'test-bg1-bgintegtest-06-01-14-20-29.cluster-ro-critvszpmydm.us-east-1.neptune.amazonaws.com', 'MultiAZ':
False, 'Engine': 'neptune', 'EngineVersion': '1.2.0.0', 'LatestRestorableTime': datetime.datetime(2023, 6, 1,
8, 51, 23, 394000, tzinfo=tzlocal()), 'Port': 8182, 'MasterUsername': 'admin', 'PreferredBackupWindow':
'06:33-07:03', 'PreferredMaintenanceWindow': 'fri:09:44-fri:10:14', 'ReadReplicaIdentifiers': [],
'DBClusterMembers': [{'DBInstanceIdentifier': 'test-bg1-bgintegtest-06-01-14-20-29i-1', 'IsClusterWriter':
True, 'DBClusterParameterGroupStatus': 'in-sync', 'PromotionTier': 1}], 'VpcSecurityGroups':
```

As mensagens de log mostram o status de sincronização entre os clusters azul e verde:

```
DDB checkpoint {'S': '1'}, {'S': '6'}

DDB Checkpoint: {'checkpointSubSequenceNumber': {'S': '6'}, 'lastUpdateTime': {'N': '1685611142127'},
'leaseOwner': {'S': 'nobody'}, 'checkpoint': {'S': '1'}, 'leaseKey': {'S': 'bg-anl   -234567899-replication'}}

Time difference for last checkpoint and last stream event: 5841351

Stream eventId difference for last replication checkpoint and last stream event on the Source cluster: 0:0

Found region : us-east-1

Cloudwatch Log Url for blue green solution is https://us-east-1.console.aws.amazon.com/cloudwatch
/home?region=us-east-1#logsV2:log-groups/log-group/aws/neptune/bg

Cloudwatch dashboard url for replication is https://console.aws.amazon.com/cloudwatch/home?region=us-
east-1#dashboards:name=neptune-stream-poller-bg-an   -234567899-replication

Replication poller lambda arn is arn:aws:lambda:us-east-1:451235071234:function:bg-an   -234567899-replic-
NeptuneStreamPollerLambd-B6V1ytULgmSP. Look for CW log the poller lambda for more troubleshooting.

Stream Last EventId {'commitNum': 1, 'opNum': 6} on cluster : database-d61852469-t        -experiment.cluster-
critvszpmydm.us-east-1.neptune.amazonaws.com:8182

DDB checkpoint {'S': '1'}, {'S': '6'}

DDB Checkpoint: {'checkpointSubSequenceNumber': {'S': '6'}, 'lastUpdateTime': {'N': '1685611207245'},
'leaseOwner': {'S': 'nobody'}, 'checkpoint': {'S': '1'}, 'leaseKey': {'S': 'bg-ankig-234567899-replication'}}
```

O processo de sincronização confere o atraso de replicação calculando a diferença entre o `eventID` do fluxo mais recente no cluster azul e o ponto de verificação de replicação presente na tabela de pontos de verificação do DynamoDB criada pela pilha de replicação Neptune para Neptune. Usando essas mensagens, é possível monitorar a diferença de replicação atual.

Passar do cluster azul de produção para o cluster verde atualizado

Antes de promover o cluster verde para produção, garanta que a diferença de confirmação entre os clusters azul e verde seja zero e, depois, desabilite todo o tráfego de gravação no cluster azul. Continuar gravando no cluster azul e alternando o endpoint do banco de dados para o cluster verde pode ocasionar a corrupção de dados causada pela gravação parcial de dados nos dois clusters. Talvez você ainda não precise desabilitar o tráfego de leitura.

Se você habilitou a autenticação do IAM no cluster de origem (azul), atualize todas as políticas do IAM usadas nas aplicações para apontar para o cluster verde (para ver um exemplo dessa política, consulte esta política de acesso irrestrito).

Depois de desabilitar o tráfego de gravação, aguarde a conclusão da replicação e habilite o tráfego de gravação no cluster verde (mas não no cluster azul). Também alterne o tráfego de leitura do cluster azul para o verde.

Limpeza após a conclusão da solução Neptune Blue/Green

Depois de promover o cluster de preparação (verde) para produção, limpe os recursos criados pela solução Neptune Blue/Green:

- Exclua a instância do Amazon EC2 que foi criada para executar a solução.
- Exclua os modelos AWS CloudFormation da replicação baseada em fluxos do Neptune que manteve o cluster verde sincronizado com o cluster azul. O principal tem o nome da pilha que você forneceu anteriormente e o outro é composto pelo ID de implantação seguida por “-replication”: ou seja, *(DeploymentID)*`-replication`.

A exclusão de modelos AWS CloudFormation não exclui os clusters em si. Depois de verificar se o cluster verde está funcionando conforme o esperado, você pode criar um snapshot antes de excluir manualmente o cluster azul.

Práticas recomendadas da solução Neptune Blue/Green

- Antes de mudar o cluster verde para produção, vale a pena verificar minuciosamente se ele está funcionando da forma correta. Confira a consistência dos dados e a configuração do banco de dados. É possível que algumas das novas versões do mecanismo também exijam atualizações do cliente. Confira as notas de versão do mecanismo antes de fazer a atualização. Vale a pena testar tudo isso em ambientes de desenvolvimento, teste e pré-produção antes de iniciar uma atualização azul/verde na produção.
- É melhor realizar a transição do servidor azul para o verde durante a janela de manutenção.
- Para garantir que tudo esteja funcionando corretamente após a atualização e a sincronização, vale a pena manter o cluster original por algum tempo antes de excluí-lo. Isso poderá ser útil se surgir um problema imprevisto.
- Evite operações de gravação pesadas, como cargas em massa, ao executar a solução Neptune Blue/Green, pois elas podem causar atrasos na replicação que introduzem um tempo de inatividade significativo. Preferencialmente, o tempo entre desativar as gravações no cluster azul e ativá-las no cluster verde deve ser de apenas alguns instantes.

Solução de problemas da solução Neptune Blue/Green

Erros gerados pela solução Neptune Blue/Green

- **`Cluster with id = (blue_green_deployment_id) already exists`**: existe um cluster com identificador *`(blue_green_deployment_id)`*.

  Forneça um novo ID de implantação ou defina o modo de implantação como `resume` se o cluster tiver sido criado em uma execução anterior do Neptune Blue/Green.
- **`Streams should be enabled on the source Cluster for Blue Green Deployment`**: ative os fluxos do Neptune no cluster azul (de origem).
- **`No Bulkload should be in progress on source cluster: (cluster_id)`**: a solução Neptune Blue/Green terminará se identificar uma carga em massa contínua.

  O objetivo é garantir que o processo de sincronização seja capaz de acompanhar as gravações feitas. Evite ou cancele qualquer trabalho de carregamento em massa contínuo antes de iniciar a solução Neptune Blue/Green.
- **`Blue Green deployment requires instances to be in sync with db cluster parameter group`**: qualquer alteração no grupo de parâmetros do cluster deve estar sincronizada em todo o cluster de banco de dados. Consulte Grupos de parâmetros do Amazon Neptune.
- **`Invalid target engine version for Blue Green Deployment`**: a versão do mecanismo de destino deve estar listada como ativa em Versões do mecanismo do Amazon Neptune e deve ser superior à versão atual do mecanismo do cluster de origem (azul).

Criar um usuário do IAM com permissões para o Neptune

Para acessar o console do Neptune para criar e gerenciar um cluster de banco de dados do Neptune, você precisa criar um usuário do IAM com todas as permissões necessárias.

A primeira etapa é criar uma política de perfil vinculado ao serviço para o Neptune:

Criar uma política de perfil vinculado ao serviço para o Amazon Neptune

1. Faça login no AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Políticas.
3. Na página Políticas, selecione Criar política.
4. Na página Criar política, selecione a guia JSON e copie a seguinte política de perfil vinculado ao serviço:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "iam:CreateServiceLinkedRole",
      "Effect": "Allow",
      "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
      "Condition": {
        "StringLike": {
            "iam:AWSServiceName":"rds.amazonaws.com"
        }
      }
    }
  ]
}
```

5. Selecione Avançar: Tags e, na página Adicionar tags, selecione Avançar: Revisão.
6. Na página Revisar política, chame a nova política de “NeptuneServiceLinked”.

Para obter mais informações sobre funções vinculadas ao serviço, consulte Usar perfis vinculados a serviços para Neptune.

Criar um usuário do IAM com todas as permissões necessárias

Depois, crie o usuário do IAM com as políticas gerenciadas apropriadas anexadas que concederão as permissões necessárias, junto com a política de perfil vinculado ao serviço que você criou (aqui denominada `NeptuneServiceLinked`):

1. Faça login no AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Usuários e, na página Usuários, selecione Adicionar usuários.
3. Na página Adicionar usuário, insira um nome para o novo usuário do IAM, selecione Chave de acesso: acesso programático para o tipo de credencial AWS e escolha Avançar: permissões.
4. Na página Definir permissões, na caixa Filtrar políticas, digite “Neptune”. Agora, selecione o seguinte nas políticas listadas:

   - NeptuneFullAccess
   - NeptuneConsoleFullAccess
   - NeptuneServiceLinked (supondo-se que seja assim que você chamou a política de perfil vinculado ao serviço criada anteriormente).
5. Depois, digite “VPC” na caixa Filtrar políticas no lugar de “Neptune”. Selecione AmazonVPCFullAccess entre as políticas listadas.
6. Selecione Avançar: Tags e, na página Adicionar tags, selecione Avançar: Revisão.
7. Na página de Revisão, confira se todas estas políticas estão agora anexadas ao novo usuário:

   - NeptuneFullAccess
   - NeptuneConsoleFullAccess
   - NeptuneServiceLinked
   - AmazonVPCFullAccess

   Depois, selecione Criar usuário.
8. Por fim, baixe e salve o ID da chave de acesso e a chave de acesso secreta do novo usuário.

Para interoperar em outros serviços, como o Amazon Simple Storage Service (Amazon S3), você precisará adicionar mais permissões e relações de confiança.

Grupos de parâmetros do Amazon Neptune

Gerencia a configuração de banco de dados no Amazon Neptune usando parâmetros em um grupo de parâmetros. Grupos de parâmetros atuam como contêineres de valores de configuração do mecanismo que são aplicados a uma ou mais instâncias de bancos de dados.

Há dois tipos de grupo de parâmetros: de cluster de banco de dados e de banco de dados.

- Os grupos de parâmetros de banco de dados aplicam-se à instância e geralmente estão associados às configurações do mecanismo de grafos do Neptune, como o parâmetro `neptune_query_timeout`.
- Os grupos de parâmetros de cluster de banco de dados aplicam-se a todas as instâncias no cluster e geralmente têm configurações mais amplas. Cada cluster do Neptune é associado a um grupo de parâmetros de cluster de banco de dados. Cada instância de banco de dados dentro desse cluster herda os valores de configuração do mecanismo contidos no grupo de parâmetros de cluster de banco de dados.

Quaisquer valores de configuração que você modificar no grupo de parâmetros de cluster de banco de dados substituirão os valores padrão no grupo de parâmetros de banco de dados. Se você editar os valores correspondentes no grupo de parâmetros de banco de dados, eles substituirão as configurações do grupo de parâmetros de cluster de banco de dados.

Um parameter group de banco de dados padrão será usado se você criar uma instância de banco de dados sem especificar um parameter group de banco de dados personalizado. Não é possível modificar as configurações de parâmetros de um grupo de parâmetros de banco de dados padrão. Em vez disso, para alterar as configurações de parâmetros padrão, é necessário criar um grupo de parâmetros de banco de dados. Nem todos os parâmetros de mecanismo de banco de dados podem ser alterados em um grupo de parâmetros de banco de dados criado por você.

Os grupos de parâmetros são criados em famílias compatíveis com diferentes versões do mecanismo do Neptune. A família de grupos de parâmetros padrão é `neptune1`, que é compatível com todas as versões do mecanismo anteriores a `1.2.0.0`. A partir de Versão: 1.2.0.0 (21/07/2022), a família do grupo de parâmetros `neptune1.2` deve ser usada em vez disso. Isso significa que, ao realizar a atualização para `1.2.0.0` ou posterior, você deve primeiro recriar todos os grupos de parâmetros personalizados na família `neptune1.2` para poder anexá-los ao realizar a atualização.

Alguns parâmetros do Neptune são estáticos e outros são dinâmicos. As diferenças são as seguintes:

Parâmetros estáticos

- Parâmetro estático é aquele que tem efeito somente após a reinicialização de uma instância de banco de dados. Em outras palavras, ao alterar um parâmetro estático e salvar o grupo de parâmetros de banco de dados da instância, é necessário reinicializar manualmente a instância de banco de dados para a alteração de parâmetros ter efeito. No momento, todos os parâmetros em nível de instância do Neptune (em um grupo de parâmetros de banco de dados em vez de em um grupo de parâmetros de cluster de banco de dados) são estáticos.
- Quando você altera um parâmetro estático no cluster e salva o grupo de parâmetros de cluster de banco de dados, a alteração do parâmetro tem efeito depois que você reinicia manualmente cada instância de banco de dados no cluster.

Parâmetros dinâmicos

- Parâmetro dinâmico é aquele que tem efeito quase imediatamente após a atualização dele no grupo de parâmetros. Em outras palavras, não há necessidade de reinicializar uma instância de banco de dados depois de atualizar um parâmetro dinâmico para que a alteração tenha efeito.
- Espere um pequeno atraso para que a alteração de um parâmetro de cluster dinâmica seja aplicada a todas as instâncias de banco de dados.
- O valor de um parâmetro dinâmico atualizado não é aplicado às solicitações em execução no momento, mas somente às enviadas após a alteração.
- Ao alterar um parâmetro dinâmico em nível de cluster, por padrão, a alteração do parâmetro é aplicada ao cluster de banco de dados imediatamente, sem a necessidade de reinicialização. Para adiar a alteração do parâmetro para depois da inicialização das instâncias de banco de dados no cluster, é possível usar a AWS CLI para definir o `ApplyMethod` como `pending-reboot` para a alteração de parâmetro.

No momento, todos os parâmetros são estáticos, exceto os seguintes novos parâmetros de cluster:

- `neptune_enable_slow_query_log` (em nível de cluster)
- `neptune_slow_query_log_threshold` (em nível de cluster)

Veja a seguir estão alguns pontos importantes que você deve saber sobre como trabalhar com parâmetros em um parameter group de banco de dados:

- Definir incorretamente os parâmetros em um parameter group de banco de dados pode causar efeitos adversos não intencionais, inclusive diminuição da performance e instabilidade no sistema. Sempre tenha cuidado ao modificar parâmetros de bancos de dados e faça backup dos dados antes de modificar um parameter group de banco de dados. Teste as alterações da configuração do parameter group em uma instância de banco de dados de teste antes de aplicar essas alterações a uma instância de banco de dados de produção.
- Ao alterar o parameter group de banco de dados associado a uma instância de banco de dados, você deve reinicializar manualmente a instância antes que o novo parameter group de banco de dados seja usado pela instância de banco de dados.

> **Note**
>
> Antes de Versão: 1.2.0.0 (21/07/2022), todas as instâncias de réplica de leitura em um cluster de banco de dados eram reinicializadas automaticamente quando a instância (de gravador) principal era reiniciada.
> De Versão: 1.2.0.0 (21/07/2022) em diante, reiniciar a instância principal não faz com que nenhuma das instâncias de réplica seja reiniciada. Isso significa que, se você estiver alterando um parâmetro em nível de cluster, deverá reiniciar cada instância separadamente para receber a alteração do parâmetro.

Edição de um grupo de parâmetros de cluster de banco de dados ou de um grupo de parâmetros de banco de dados

1. Faça login no Console de Gerenciamento da AWS e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. Selecione Parameter groups no painel de navegação.
3. Selecione o link Name (Nome) para o parameter group de banco de dados que você deseja editar.

   (Opcional) Selecione Create parameter group (Criar parameter group) para criar um novo parameter group de cluster e crie o novo grupo. Em seguida, selecione o Name (Nome) do novo parameter group.

> **Important**
>
> Essa etapa será necessária se você tiver apenas o parameter group padrão de cluster de banco de dados porque o parameter group padrão de cluster de banco de dados não pode ser modificado.

4. Pesquise o parâmetro e clique no campo Valor ao lado da coluna Nome.
5. Insira o valor permitido e escolha a marca ao lado do campo de valor.
6. Escolha Salvar alterações.
7. Reinicialize cada instância de banco de dados no cluster do Neptune se você estiver alterando um parâmetro de cluster de banco de dados, ou uma ou mais instâncias específicas se estiver alterando um parâmetro de instância de banco de dados.

Criar um grupo de parâmetros de banco de dados ou um grupo de parâmetros de cluster de banco de dados

É possível usar com facilidade o console do Neptune para criar um grupo de parâmetros:

1. Faça login no Console de Gerenciamento da AWS e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. Escolha Parameter Groups no painel de navegação à esquerda.
3. Escolha Create parameter group de banco de dados (Criar parameter group de banco de dados).

   A página Create parameter group de banco de dados (Criar parameter group de banco de dados) é exibida.
4. Na lista Família de grupos de parâmetros, escolha neptune1 ou, se você estiver visando a versão 1.2.0.0 ou posterior do mecanismo, escolha neptune1.2.
5. Na lista Type (Tipo), escolha DB Parameter Group (Parameter group de banco de dados) ou DB Cluster Parameter Group (Parameter group de cluster de banco de dados).
6. Digite o nome do novo parameter group de banco de dados na caixa Group name (Nome do grupo).
7. Digite uma descrição para o novo parameter group de banco de dados na caixa Description (Descrição).

8. Escolha Criar.

Também é possível criar um grupo de parâmetros usando AWS CLI:

```
aws neptune create-db-parameter-group \
  --db-parameter-group-name (a name for the new DB parameter group) \
  --db-parameter-group-family (either neptune1 or neptune1.2, depending on the engine
 version) \
  --description (a description for the new DB parameter group)
```

Parâmetros do Amazon Neptune

Gerencia a configuração de banco de dados no Amazon Neptune usando parâmetros em grupo de parâmetros. Os seguintes parâmetros estão disponíveis para configurar o banco de dados do Neptune:

Parâmetros no nível do cluster

- neptune_enable_audit_log
- neptune_enable_slow_query_log
- neptune_slow_query_log_threshold
- neptune_lab_mode
- neptune_query_timeout
- neptune_streams
- neptune_streams_expiry_days
- neptune_lookup_cache
- neptune_autoscaling_config
- neptune_ml_iam_role
- neptune_ml_endpoint

Parâmetros no nível da instância

- neptune_dfe_query_engine
- neptune_query_timeout
- neptune_result_cache

Parâmetros obsoletos

- neptune_enforce_ssl

`neptune_enable_audit_log` (parâmetro em nível de cluster)

Esse parâmetro alterna o registro em log de auditoria do Neptune.

Os valores permitidos são `0` (desabilitado) e `1` (habilitado). O valor padrão é `0`.

Esse parâmetro é estático, o que significa que as alterações nele não têm efeito em nenhuma instância até que ela seja reinicializada.

É possível publicar logs de auditoria no Amazon CloudWatch, conforme descrito em Usando a CLI para publicar registros de auditoria do Neptune no Logs CloudWatch .

`neptune_enable_slow_query_log` (parâmetro em nível de cluster)

Use esse parâmetro para habilitar ou desabilitar o atributo de registro em log de consultas lentas do Neptune.

Esse é um parâmetro dinâmico, o que significa que a alteração do valor não exige nem causa a reinicialização do cluster de banco de dados.

Os valores permitidos são:

- **`info`**: permite o registro em log de consultas lentas e registra os atributos selecionados que podem estar contribuindo para o baixo desempenho.
- **`debug`**: permite o registro em log de consultas lentas e registra todos os atributos disponíveis da execução da consulta.
- **`disable`**: desabilita o registro em log de consultas lentas.

O valor padrão é `disable`.

É possível publicar logs de consultas lentas no Amazon CloudWatch, conforme descrito em Usando a CLI para publicar registros de consulta lenta do Neptune no Logs CloudWatch .

`neptune_slow_query_log_threshold` (parâmetro em nível de cluster)

Esse parâmetro especifica o limite de tempo de execução, em milissegundos, após o qual uma consulta é considerada lenta. Se o registro em log de consultas lentas estiver habilitado, as consultas executadas acima desse limite serão registradas junto com alguns dos atributos.

O valor padrão é cinco mil milissegundos (cinco segundos).

Esse é um parâmetro dinâmico, o que significa que a alteração do valor não exige nem causa a reinicialização do cluster de banco de dados.

`neptune_lab_mode` (parâmetro em nível de cluster)

Quando definido, esse parâmetro habilita atributos experimentais específicos do Neptune. Consulte Modo de laboratório do Neptune para verificar os recursos experimentais disponíveis no momento.

Esse parâmetro é estático, o que significa que as alterações nele não têm efeito em nenhuma instância até que ela seja reinicializada.

Para habilitar ou desabilitar um recurso experimental, inclua *(nome do recurso)*=enabled ou *(nome do recurso)*=disabled neste parâmetro. Você pode habilitar ou desabilitar vários recursos separando-os com vírgulas, como este:

(nome do recurso 1)=enabled,*(nome do recurso 2)*=enabled

Os atributos do modo de laboratório geralmente são desabilitados por padrão. Uma exceção é o atributo DFEQueryEngine, que foi habilitado por padrão para uso com dicas de consulta (DFEQueryEngine=viaQueryHint) a partir da versão 1.0.5.0 do mecanismo do Neptune. A partir da versão 1.1.1.0 do mecanismo do Neptune, o mecanismo DFE não está mais no modo de laboratório e agora é controlado usando o parâmetro de instância neptune_dfe_query_engine no grupo de parâmetros de banco de dados de uma instância.

`neptune_query_timeout` (parâmetro em nível de cluster)

Especifica uma duração de tempo limite específica para consultas de grafo, em milissegundos.

Os valores permitidos variam de 10 a 2,147,483,647 (2^{31} - 1). O valor padrão é 120,000 (2 minutos).

Esse parâmetro é estático, o que significa que as alterações nele não têm efeito em nenhuma instância até que ela seja reinicializada.

> **Note**
>
> Você poderá gerar custos inesperados se definir um valor de tempo limite de consulta muito alto, especialmente em uma instância sem servidor. Sem uma configuração de

tempo limite razoável, você poderá emitir acidentalmente uma consulta que continue sendo executada por muito mais tempo do que o esperado, gerando custos jamais previstos. Isso é particularmente o caso em uma instância sem servidor cuja escala pode ser aumentada verticalmente para um tipo de instância grande e caro durante a execução da consulta. É possível evitar despesas inesperadas desse tipo usando um valor de tempo limite de consulta que acomode a maioria das consultas e ocasione apenas um tempo limite de execução excepcionalmente longo.

`neptune_streams` (parâmetro em nível de cluster)

Habilita ou desabilita o Fluxos do Neptune.

Esse parâmetro é estático, o que significa que as alterações nele não têm efeito em nenhuma instância até que ela seja reinicializada.

Os valores permitidos são `0` (desabilitado, que é o padrão), e `1` (habilitado).

`neptune_streams_expiry_days` (parâmetro em nível de cluster)

Especifica quantos dias decorrem antes que o servidor exclua os registros de fluxo.

Os valores permitidos são de `1` a `90`. O padrão é `7`.

Esse parâmetro foi apresentado na versão 1.2.0.0 do mecanismo.

Esse parâmetro é estático, o que significa que as alterações nele não têm efeito em nenhuma instância até que ela seja reinicializada.

`neptune_lookup_cache` (parâmetro em nível de cluster)

Desabilita ou reabilita o cache de pesquisa do Neptune em instâncias `R5d`.

Esse parâmetro é estático, o que significa que as alterações nele não têm efeito em nenhuma instância até que ela seja reinicializada.

Os valores permitidos são `enabled` e `disabled`. O valor padrão é `disabled`, mas sempre que uma instância `R5d` é criada no cluster de banco de dados, o parâmetro `neptune_lookup_cache` é automaticamente definido como `enabled` e um cache de pesquisa é criado nessa instância.

`neptune_autoscaling_config` (parâmetro em nível de cluster)

Define parâmetros de configuração para as instâncias de réplica de leitura que o ajuste de escala automático do Neptune cria e gerencia.

Esse parâmetro é estático, o que significa que as alterações nele não têm efeito em nenhuma instância até que ela seja reinicializada.

Usando uma string JSON que você define como o valor do parâmetro `neptune_autoscaling_config`, é possível especificar:

- O tipo de instância que o ajuste de escala automático do Neptune usa para todas as novas instâncias de réplica de leitura que ele cria.
- As janelas de manutenção atribuídas a essas réplicas de leitura.
- Tags a serem associadas a todas as novas réplicas de leitura.

A string JSON tem uma estrutura como esta:

```
"{
  \"tags\": [
    { \"key\" : \"reader tag-0 key\", \"value\" : \"reader tag-0 value\" },
    { \"key\" : \"reader tag-1 key\", \"value\" : \"reader tag-1 value\" },
  ],
  \"maintenanceWindow\" : \"wed:12:03-wed:12:33\",
  \"dbInstanceClass\" : \"db.r5.xlarge\"
}"
```

Observe que as aspas dentro da string devem ter todas um caractere de escape que é uma barra invertida (\).

Qualquer uma das três definições de configuração não especificadas no parâmetro `neptune_autoscaling_config` é copiada da configuração da instância de gravador principal do cluster de banco de dados.

`neptune_ml_iam_role` (parâmetro em nível de cluster)

Especifica o ARN do perfil do IAM usado no Neptune ML. O valor pode ser qualquer ARN de perfil do IAM válido.

Esse parâmetro é estático, o que significa que as alterações nele não têm efeito em nenhuma instância até que ela seja reinicializada.

É possível especificar o ARN do perfil do IAM padrão para machine learning em grafos.

`neptune_ml_endpoint` (parâmetro em nível de cluster)

Especifica o endpoint usado para o Neptune ML. O valor pode ser qualquer nome de endpoint do SageMaker válido.

Esse parâmetro é estático, o que significa que as alterações nele não têm efeito em nenhuma instância até que ela seja reinicializada.

É possível especificar o endpoint padrão do SageMaker para machine learning em grafos.

`neptune_dfe_query_engine` (parâmetros em nível de instância)

A partir da versão 1.1.1.0 do mecanismo do Neptune, esse parâmetro de instância de banco de dados é usado para controlar como o mecanismo de consulta do DFE é usado. Os valores permitidos são os seguintes:

Esse parâmetro é estático, o que significa que as alterações nele não têm efeito em nenhuma instância até que ela seja reinicializada.

- **`enabled`**: faz com que o mecanismo DFE seja usado sempre que possível, exceto quando a dica de consulta `useDFE` está presente e definida como `false`.
- **`viaQueryHint`** (o padrão): faz com que o mecanismo DFE seja usado somente para consultas que incluam explicitamente a dica de consulta `useDFE` definida como `true`.

Se esse parâmetro não tiver sido definido explicitamente, o valor padrão, `viaQueryHint`, será usado quando a instância for iniciada.

> **Note**
>
> Todas as consultas do openCypher são executadas pelo mecanismo DFE, independentemente de como esse parâmetro é definido.

Antes da versão 1.1.1.0, era um parâmetro de modo de laboratório em vez de um parâmetro de instância de banco de dados.

`neptune_query_timeout` (parâmetros em nível de instância)

Esse parâmetro de instância de banco de dados especifica uma duração de tempo limite para consultas de grafos, em milissegundos, para uma instância.

Esse parâmetro é estático, o que significa que as alterações nele não têm efeito em nenhuma instância até que ela seja reinicializada.

Os valores permitidos variam de `10` a `2,147,483,647` (2^{31} - 1). O valor padrão é `120,000` (2 minutos).

> **Note**
>
> Você poderá gerar custos inesperados se definir um valor de tempo limite de consulta muito alto, especialmente em uma instância sem servidor. Sem uma configuração de tempo limite razoável, você poderá emitir acidentalmente uma consulta que continue sendo executada por muito mais tempo do que o esperado, gerando custos jamais previstos. Isso é particularmente o caso em uma instância sem servidor cuja escala pode ser aumentada verticalmente para um tipo de instância grande e caro durante a execução da consulta. É possível evitar despesas inesperadas desse tipo usando um valor de tempo limite de consulta que acomode a maioria das consultas e ocasione apenas um tempo limite de execução excepcionalmente longo.

`neptune_result_cache` (parâmetros em nível de instância)

`neptune_result_cache`: esse parâmetro de instância de banco de dados habilita ou desabilita Armazenar em cache os resultados da consulta.

Esse parâmetro é estático, o que significa que as alterações nele não têm efeito em nenhuma instância até que ela seja reinicializada.

Os valores permitidos são `0` (desabilitado, que é o padrão) e 1 (habilitado).

`neptune_enforce_ssl` (parâmetro em nível de cluster obsoleto)

(Obsoleto) Costumava haver regiões que permitiam conexões HTTP com o Neptune, e esse parâmetro era usado para forçar todas as conexões a usarem HTTPS quando definido como 1. No

entanto, esse parâmetro não é mais relevante, pois o Neptune agora só aceita conexões HTTPS em todas as regiões.

Iniciar um cluster de banco de dados do Neptune usando o AWS Management Console

A maneira mais fácil de iniciar um novo cluster de banco de dados do Neptune é usar um modelo AWS CloudFormation que crie todos os recursos necessários para você, conforme explicado em Criar um cluster de banco de dados.

Se preferir, você também pode usar o console do Neptune para iniciar um novo cluster de banco de dados manualmente, conforme explicado aqui.

Para acessar o console do Neptune para criar e gerenciar um cluster do Neptune, crie um usuário do IAM com todas as permissões necessárias para fazê-lo, conforme explicado em Criar um usuário do IAM com permissões para o Neptune.

Depois, faça login no AWS Management Console como esse usuário do IAM e siga as etapas abaixo para criar um cluster de banco de dados:

Como iniciar um cluster de banco de dados do Neptune usando o console

1. Faça login no Console de Gerenciamento da AWS e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. Acesse a página Bancos de dados e escolha Criar banco de dados, que abre a página Criar banco de dados.
3. Em Opções de mecanismo, o tipo de mecanismo é `neptune`, e você pode escolher uma versão específica do mecanismo ou aceitar o padrão.
4. Em Configurações, insira um nome para o novo cluster de banco de dados ou aceite o nome padrão fornecido. Esse nome é usado no endereço do endpoint da instância e deve atender às seguintes restrições:
   - Deve conter de 1 a 63 caracteres alfanuméricos ou hífens.
   - O primeiro caractere deve ser uma letra.
   - Não pode terminar com um hífen ou conter dois hifens consecutivos.
   - Deve ser exclusivo para todas as instâncias de banco de dados na conta da AWS em uma região da AWS específica.
5. Em Modelos, escolha Produção ou Desenvolvimento e teste.

6. Em Tamanho da instância de banco de dados, escolha um tamanho de instância. Isso determinará o processamento e a capacidade da memória da instância de gravação principal do novo cluster de banco de dados.

   Se você selecionou o modelo de produção, só poderá escolher entre as classes disponíveis com otimização de memória listadas, mas se tiver selecionado Desenvolvimento e teste, também poderá escolher entre as classes com capacidade de intermitência mais econômicas (consulte Instâncias de intermitência T3 para ver uma discussão sobre classes com capacidade de intermitência).

   > **Note**
   >
   > A partir da versão 1.1.0.0 do mecanismo do Neptune, o Neptune não é mais compatível com tipos de instância R4.

7. Em Disponibilidade e durabilidade, é possível escolher se deseja ou não habilitar a implantação de várias zonas de disponibilidade (multi-AZ). O modelo de produção permite a implantação multi-AZ por padrão, enquanto o modelo de desenvolvimento e teste não. Se a implantação multi-AZ estiver habilitada, o Neptune localizará instâncias de réplica de leitura criadas em diferentes zonas de disponibilidade (AZs) para melhorar a disponibilidade.
8. Em Conectividade, selecione a nuvem privada virtual (VPC) que hospedará o novo cluster de banco de dados dentre as opções disponíveis. Aqui é possível escolher Criar VPC se quiser que o Neptune crie a VPC. Você deve criar uma instância do Amazon EC2 nessa mesma VPC para acessar a instância do Neptune (para obter mais informações, consulte Cada cluster de banco de dados do Amazon Neptune reside em uma Amazon VPC). Observe que não é possível alterar a VPC após a criação do cluster de banco de dados.

   Se precisar, você poderá configurar ainda mais a conectividade do cluster em Configuração de conectividade adicional:

   a. Em Grupo de sub-redes, você pode escolher o grupo de sub-redes de banco de dados do Neptune a ser usado para o novo cluster de banco de dados. Se a VPC ainda não tiver grupos de sub-redes, o Neptune criará um grupo de sub-redes de banco de dados para você (consulte Cada cluster de banco de dados do Amazon Neptune reside em uma Amazon VPC).
   b. Em Grupos de segurança da VPC, selecione um ou mais grupos de segurança da VPC existentes para proteger o acesso à rede ao novo cluster de banco de dados ou escolha Criar novo se quiser que o Neptune crie um para você e, depois, forneça um nome para o

novo grupo de segurança da VPC (consulte Criar um grupo de segurança usando o console da VPC).

c. Em Porta do banco de dados, insira a porta TCP/IP que o banco de dados usará para conexões de aplicações. O Neptune usa o número da porta 8182 como padrão.

9. Em Configuração do bloco de anotações, escolha Criar bloco de anotações se quiser que o Neptune crie cadernos Jupyter para você na bancada de trabalho do Neptune (consulte Usar os blocos de anotações de grafos Neptune para começar rapidamente e Usar a bancada de trabalho do Neptune para hospedar blocos de anotações Neptune). Depois, você pode escolher como os novos blocos de anotações devem ser configurados:

   a. Em Tipo de instância de bloco de anotações, escolha entre as classes de instância disponíveis para o bloco de anotações.

   b. Em Nome do bloco de anotações, insira um nome para o bloco de anotações.

   c. Se quiser, você também poderá inserir uma descrição do bloco de anotações em Descrição: opcional.

   d. Em Nome do perfil do IAM, escolha fazer com que o Neptune crie um perfil do IAM para o bloco de anotações e insira um nome para o novo perfil, ou escolha selecionar um perfil do IAM existente dentre os perfis disponíveis.

   e. Por fim, escolha se o bloco de anotações se conecta à Internet diretamente ou por meio do Amazon SageMaker ou por meio de uma VPC com um gateway NAT. Consulte Conectar uma instância do bloco de anotações aos recursos em uma VPC para obter mais informações.

10. Em Tags, é possível associar até cinquenta tags ao novo cluster de banco de dados.

11. Em Configuração adicional, há mais configurações que você pode definir para o novo cluster de banco de dados (em muitos casos, você pode ignorá-las e aceitar os valores padrão por enquanto):

Opção	O que é possível fazer
DB instance identifier	É possível fornecer um nome para a instância de gravador do cluster. Caso contrário, um identificador padrão baseado no nome do cluster será usado. Em caso afirmativo, especifique um nome exclusivo para todas as instâncias de banco de dados

Opção	O que é possível fazer
	pertencentes à conta da AWS na região atual. O identificador de instância de banco de dados não diferencia letras maiúsculas de minúsculas, mas é armazenado com todas as letras minúsculas.
Grupo de parâmetros do cluster de banco de dados	Selecione um grupo de parâmetros de cluster de banco de dados para definir a configuração padrão para todas as instância s de banco de dados no cluster. A menos que você escolha o contrário, o Neptune usa um grupo de parâmetros de cluster de banco de dados padrão. Para ter mais informações sobre parameter groups, consulte Grupos de parâmetros do Amazon Neptune.
Grupo de parâmetros de banco de dados	Selecione um grupo de parâmetros de banco de dados para definir a configuração da instância de banco de dados principal no cluster. A menos que você escolha o contrário, o Neptune usa um grupo de parâmetros padrão. Para ter mais informaçõ es sobre parameter groups, consulte Grupos de parâmetros.

Opção	O que é possível fazer
IAM DB authentication	Se você selecionar Habilitar autenticação do banco de dados do IAM, todo o acesso ao banco de dados será autenticado usando o AWS Identity and Access Management (IAM). **Important** Isso requer que você assine todas as solicitações com o AWS Signature versão 4. Para obter mais informaçõ es, consulte Visão geral do AWS Identity and Access Management (IAM) no Amazon Neptune.
Prioridade de failover	Escolha `No preference` ou uma camada de prioridade para failover. Se você escolher uma cada e houver contenção nela, a réplica que tem o mesmo tamanho da instância principal será selecionada.
Backup retention period (Período de retenção de backup)	Escolha o tempo, de 1 a 35 dias, durante o qual o Neptune deve reter backups automátic os dessa instância de banco de dados. Você só pode realizar uma restauração para um ponto no tempo (PITR) dentro do período de retenção de backup.
Copiar tags para snapshots	(Habilitada por padrão) Essa opção faz com que todas as tags associadas ao cluster de banco de dados sejam copiadas em qualquer snapshot dele.

Opção	O que é possível fazer
Habilitar criptografia	(Habilitada por padrão) Essa opção faz com que os dados no cluster de banco de dados sejam criptografados em repouso. Em caso afirmativo, escolha a chave mestra usada para proteger a chave utilizada para criptografar esse volume de banco de dados. É possível selecionar a chave `aws/rds` padrão, escolha entre as chaves mestras na conta ou insira o ARN de uma chave de outra conta. É possível criar uma chave mestra de criptografia acessando a guia Chaves de criptografia do console do IAM. Para obter mais informações, consulte Criptografia de recursos em repouso do Neptune.
Log de auditoria	Marque se quiser que os logs de auditoria do cluster de banco de dados sejam publicados no CloudWatch Logs.
Habilitar a atualização automática da versão secundária	(Habilitada por padrão) Essa opção faz com que o cluster de banco de dados seja atualizado automaticamente para novas versões secundárias do mecanismo após o lançamento. As atualizações automátic as ocorrem durante a janela de manutençã o do banco de dados. Consulte Usar o `AutoMinorVersionUpgrade` .

Opção	O que é possível fazer
Janela de manutenção	É possível selecionar um período específico durante o qual deseja que as modificações pendentes no cluster de banco de dados ocorram, como uma alteração em uma classe de instância de banco de dados ou um patch automático do mecanismo. Todas essas operações de manutenção são iniciadas e concluídas no período selecionado. Se você não selecionar um período, o Neptune atribuirá um período de manutenção arbitrariamente.
Habilitar proteção contra exclusão	Habilitada por padrão A proteção contra exclusão impede que o cluster de banco de dados seja excluído. É necessário desabilitá-la explicitamente para excluir o cluster de banco de dados.

12. Escolha Criar banco de dados para iniciar o novo cluster de banco de dados do Neptune e a instância principal.

    No console do Amazon Neptune, o novo cluster de banco de dados é exibido na lista de banco de dados. O cluster de banco de dados tem o status Creating (Criando) até que esteja criado e pronto para uso. Quando o status for alterado para Available (Disponível), você poderá se conectar à instância primária do seu cluster de banco de dados. Dependendo da classe da instância de banco de dados e do armazenamento alocado, pode levar alguns minutos até que novas instâncias fiquem disponíveis.

    Para visualizar o cluster recém-criado, selecione a exibição de Bancos de dados no console do Neptune.

    > **Note**
    >
    > Se você excluir todas as instâncias de banco de dados do Neptune em um cluster de banco de dados usando o AWS Management Console, o console excluirá automaticamente o próprio cluster de banco de dados. Se estiver usando a AWS CLI ou

o SDK, você deverá excluir o cluster de banco de dados manualmente depois de excluir a última instância.

Anote o valor do Endpoint do cluster. Você precisará dele para conectar-se ao cluster de banco de dados do Neptune.

Interromper e iniciar um cluster de banco de dados do Amazon Neptune

Interromper e iniciar os clusters do Amazon Neptune ajuda a gerenciar os custos dos ambientes de teste e de desenvolvimento. Você pode interromper temporariamente todas as instâncias de banco de dados em seu cluster, em lugar de configurar e destruir todas as instâncias cada vez que você usa o cluster.

Tópicos

Visão geral de como interromper e iniciar um cluster de banco de dados do Neptune

Durante os períodos em que você não precisa de um cluster do Neptune, é possível interromper todas as instâncias nesse cluster de uma só vez. Você pode iniciar o cluster novamente a qualquer momento, sempre que precisar usá-lo. Iniciar e interromper simplifica os processos de configuração e destruição dos clusters usados em desenvolvimento, teste ou atividades afins que não exijam disponibilidade contínua. É possível fazer isso no AWS Management Console com uma única ação, independentemente de quantas instâncias existam no cluster.

Durante a interrupção do cluster de banco de dados, serão cobrados somente o armazenamento do cluster, os snapshots manuais e o armazenamento do backup automático dentro da janela de retenção especificada. Não haverá cobrança por horas de instância de banco de dados.

Após sete dias, o Neptune iniciará automaticamente o cluster de banco de dados novamente para garantir que ele não perca nenhuma atualização de manutenção necessária.

Para minimizar as cobranças em um cluster do Neptune levemente carregado, é possível interromper o cluster em vez de excluir todas as réplicas de leitura dele. Para clusters com mais de uma ou duas instâncias, excluir e recriar as instâncias de banco de dados com frequência só é prático usando a AWS CLI ou a API do Neptune, e também pode ser difícil realizar as exclusões na ordem correta. Por exemplo, você deve excluir todas as réplicas de leitura antes de excluir a instância principal para evitar que o mecanismo de failover seja ativado.

Não use a interrupção e a inicialização se precisar manter o cluster de banco de dados em execução, mas quiser reduzir a capacidade. Se o cluster for muito caro ou não estiver muito ocupado, você poderá excluir uma ou mais instâncias de banco de dados ou alterar suas instâncias de banco de dados para usar uma classe de instância menor, mas você não poderá interromper uma instância de banco de dados individual.

Interromper um cluster de banco de dados do Neptune

Quando você não estiver usando um cluster de banco de dados do Neptune por algum tempo, é possível interromper sua execução e iniciá-lo novamente quando precisar dele. Enquanto o cluster estiver parado, serão cobrados o armazenamento do cluster, os snapshots manuais e o armazenamento de backup automático dentro da janela de retenção especificada. As horas de instância de banco de dados não serão cobradas.

A operação de interrupção interrompe todas as instâncias de réplica de leitura do cluster antes de interromper a instância primária para evitar a ativação do mecanismo de failover.

Interromper um cluster de banco de dados usando o AWS Management Console

Como usar o AWS Management Console para interromper um cluster do Neptune

1. Faça login no Console de Gerenciamento da AWS e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. No painel de navegação, escolha Bancos de dados e escolha um cluster. Você pode executar a operação de interrupção nesta página ou navegar até a página de detalhes do cluster de banco de dados que você deseja interromper.
3. Em Actions (Ações), escolha Stop (Interromper).

Interromper um cluster de banco de dados usando o AWS CLI

Para interromper uma instância de banco de dados usando a AWS CLI, chame o comando stop-db-cluster usando o parâmetro `--db-cluster-identifier` para identificar o cluster de banco de dados que deseja interromper.

Example

```
aws neptune stop-db-cluster --db-cluster-identifier mydbcluster
```

Interromper um cluster de banco de dados usando a API de gerenciamento do Neptune

Para interromper uma instância de banco de dados usando a API de gerenciamento do Neptune, chame a API StopDBCluster e use o parâmetro `DBClusterIdentifier` para identificar o cluster de banco de dados que deseja interromper.

O que pode acontecer enquanto um cluster de banco de dados está parado

- É possível restaurá-lo a partir de um snapshot (consulte Restaurar a partir de um snapshot de cluster de banco de dados).
- Não é possível modificar a configuração do cluster de banco de dados ou de qualquer uma de suas instâncias de banco de dados.
- Não é possível adicionar ou remover instâncias de banco de dados do cluster.
- Não é possível excluir o cluster se ele ainda tiver instâncias de banco de dados associadas.
- Em geral, você deve reiniciar um cluster de banco de dados interrompido para executar a maioria das ações administrativas.
- O Neptune aplicará as manutenções programadas no cluster interrompido assim que ele for reiniciado. Lembre-se de que, após sete dias, o Neptune reiniciará automaticamente um cluster interrompido para não atrasar demais o status de manutenção.
- O Neptune não executa o backup automatizado de um cluster de banco de dados interrompido, pois os dados subjacentes não podem ser alterados enquanto o cluster está parado.
- O Neptune não estende o período de retenção de backup para o cluster de banco de dados enquanto ele está parado.

Iniciar um cluster de banco de dados do Neptune interrompido

Só é possível iniciar um cluster de banco de dados do Neptune que esteja no estado interrompido. Quando você inicia o cluster, todas as suas instâncias de banco de dados ficam disponíveis novamente. O cluster mantém suas configurações, como endpoints, grupos de parâmetros e grupos de segurança da VPC.

Iniciar um cluster de banco de dados interrompido usando o AWS Management Console

1. Faça login no Console de Gerenciamento da AWS e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. No painel de navegação, escolha Bancos de dados e escolha um cluster. É possível executar a operação de inicialização nesta página ou navegar até a página de detalhes do cluster de banco de dados e fazer a inicialização nela.
3. Em Actions (Ações), escolha Start (Iniciar).

Iniciar um cluster de banco de dados interrompido usando o AWS CLI

Para iniciar um cluster de banco de dados interrompido usando a AWS CLI, chame o comando start-db-cluster usando o parâmetro `--db-cluster-identifier` para especificar o cluster de banco de dados interrompido que deseja iniciar. Forneça o nome do cluster que você escolheu ao criar o cluster de banco de dados ou use um nome de instância de banco de dados que você escolheu, com `-cluster` anexado ao final dele.

Example

```
aws neptune start-db-cluster --db-cluster-identifier mydbcluster
```

Iniciar um cluster de banco de dados interrompido usando a API de gerenciamento do Neptune

Para iniciar um cluster de banco de dados do Neptune usando a API de gerenciamento do Neptune, chame a API StartDBCluster usando o parâmetro `DBCluster` para especificar o cluster de banco de dados interrompido que deseja iniciar. Forneça o nome do cluster que você escolheu ao criar o cluster de banco de dados ou use um nome de instância de banco de dados que você escolheu, com `-cluster` anexado ao final dele.

Esvaziar um cluster de banco de dados do Amazon Neptune usando a API de redefinição rápida

A API REST de redefinição rápida do Neptune permite redefinir um grafo do Neptune com rapidez e facilidade, removendo todos seus dados.

É possível fazê-lo em um bloco de anotações Neptune usando a magia de linha %db_reset.

> **Note**
>
> Esse atributo está disponível a partir da versão 1.0.4.0 do mecanismo do Neptune.

- Na maioria dos casos, uma operação de redefinição rápida é concluída em alguns minutos. A duração pode variar um pouco dependendo da carga no cluster quando a operação é iniciada.
- Uma operação de redefinição rápida não gera E/S adicionais.
- O tamanho do volume de armazenamento não diminui após uma redefinição rápida. Em vez disso, o armazenamento é reutilizado à medida que novos dados são inseridos. Isso significa que os tamanhos dos volumes dos snapshots criados antes e depois de uma operação de redefinição rápida serão os mesmos. Os tamanhos dos volumes dos clusters restaurados usando os snapshots criados antes e depois de uma operação de redefinição rápida também serão os mesmos
- Como parte da operação de redefinição, todas as instâncias no cluster de banco de dados são reiniciadas.

> **Note**
>
> Em raras condições, essas redefinições do servidor também podem ocasionar failover do cluster.

> **Important**
>
> Usar a redefinição rápida pode interromper a integração do cluster de banco de dados do Neptune com outros serviços. Por exemplo:

- A redefinição rápida exclui todos os dados de fluxo do banco de dados e redefine completamente os fluxos. Isso significa que os consumidores de fluxos podem não funcionar mais sem uma nova configuração.
- A redefinição rápida remove todos os metadados sobre os recursos do SageMaker que estão sendo usados pelo Neptune ML, incluindo trabalhos e endpoints. Eles continuam existindo no SageMaker, e você pode continuar usando os endpoints existentes do SageMaker para consultas de inferência do Neptune ML, mas as APIs de gerenciamento do Neptune ML não funcionam mais com eles.
- Integrações, como a integração de pesquisa de texto completo com o ElasticSearch, também são eliminadas pela redefinição rápida e devem ser restabelecidas manualmente para poderem ser usadas novamente.

Como excluir todos os dados de um cluster de banco de dados do Neptune usando a API

1. Primeiro, você gera um token que pode ser usado para realizar a redefinição do banco de dados. Essa etapa tem como objetivo ajudar a impedir que redefinam acidentalmente um banco de dados.

   Você faz isso enviando uma solicitação `HTTP POST` ao endpoint `/system` na instância de gravador do cluster de banco de dados para especificar a ação `initiateDatabaseReset`.

   O comando `curl` usando o tipo de conteúdo JSON seria:

   ```
   curl -X POST \
     -H 'Content-Type: application/json' \
         https://your_writer_instance_endpoint:8182/system \
     -d '{ "action" : "initiateDatabaseReset" }'
   ```

   Ou, usando o tipo de conteúdo `x-www-form-urlencoded`:

   ```
   curl -X POST \
     -H 'Content-Type: application/x-www-form-urlencoded' \
         https://your_writer_instance_endpoint:8182/system \
     -d 'action=initiateDatabaseReset '
   ```

A solicitação `initiateDatabaseReset` exibe o token de redefinição na resposta JSON, da seguinte forma:

```
{
  "status" : "200 OK",
  "payload" : {
    "token" : "new_token_guid"
  }
}
```

O token permanece válido por uma hora (sessenta minutos) após a emissão.

Se você enviar a solicitação a uma instância de leitor ou ao endpoint de status, o Neptune lançará um `ReadOnlyViolationException`.

Se você enviar várias solicitações `initiateDatabaseReset`, somente o token mais recente gerado será válido para a segunda etapa, na qual você realmente executará a redefinição.

Se o servidor for reiniciado logo após a solicitação `initiateDatabaseReset`, o token gerado se tornará inválido e você precisará enviar uma nova solicitação para obter um novo token.

2. Depois, você envia uma solicitação `performDatabaseReset` com o token recebido de volta de `initiateDatabaseReset` ao endpoint `/system` na instância de gravador do cluster de banco de dados. Isso exclui todos os dados do cluster de banco de dados.

   O comando `curl` usando o tipo de conteúdo JSON é:

   ```
   curl -X POST \
     -H 'Content-Type: application/json' \
         https://your_writer_instance_endpoint:8182/system \
     -d '{
           "action" : "performDatabaseReset",
           "token" : "token_guid"
         }'
   ```

   Ou, usando o tipo de conteúdo `x-www-form-urlencoded`:

   ```
   curl -X POST \
     -H 'Content-Type: application/x-www-form-urlencoded' \
         https://your_writer_instance_endpoint:8182/system \
   ```

```
  -d 'action=performDatabaseReset&token=token_guid'
```

A solicitação gera uma resposta JSON. Se a solicitação for aceita, a resposta será:

```
{
  "status" : "200 OK"
}
```

Se o token que você enviou não corresponder ao que foi emitido, a resposta será a seguinte:

```
{
  "code" : "InvalidParameterException",
  "requestId":"token_guid",
  "detailedMessage" : "System command parameter 'token' : 'token_guid' does not
 match database reset token"
}
```

Se a solicitação for aceita e a redefinição começar, o servidor será reiniciado e excluirá os dados. Você não pode enviar nenhuma outra solicitação ao cluster de banco de dados enquanto ele está sendo redefinido.

Usar a API de redefinição rápida com IAM-Auth

Se você tiver a IAM-Auth habilitada no cluster de banco de dados, poderá usar o awscurl para enviar comandos de redefinição rápida que são autenticados usando a IAM-Auth:

Usar o awscurl para enviar solicitações de redefinição rápida com IAM-Auth

1. Defina as variáveis de ambiente `AWS_ACCESS_KEY_ID` e `AWS_SECRET_ACCESS_KEY` corretamente (e também `AWS_SECURITY_TOKEN` se você estiver usando uma credencial temporária).
2. Uma solicitação `initiateDatabaseReset` tem a seguinte aparência:

```
awscurl -X POST --service neptune-db "$SYSTEM_ENDPOINT" \
  -H 'Content-Type: application/json' --region us-west-2 \
  -d '{ "action" : "initiateDatabaseReset" }'
```

3. Uma solicitação `performDatabaseReset` tem a seguinte aparência:

```
awscurl -X POST --service neptune-db "$SYSTEM_ENDPOINT" \
  -H 'Content-Type: application/json' --region us-west-2 \
  -d '{ "action" : "performDatabaseReset" }'
```

Usar a magia de linha **`%db_reset`** da bancada de trabalho do Neptune para redefinir um cluster de banco de dados

A bancada de trabalho do Neptune é compatível com uma magia de linha `%db_reset` que permite realizar uma redefinição rápida do banco de dados em um bloco de anotações Neptune.

Se você invocar a magia sem nenhum parâmetro, verá uma tela perguntando se deseja excluir todos os dados do cluster, com uma caixa de seleção solicitando que você reconheça que os dados do cluster não estarão mais disponíveis após a exclusão. Nesse ponto, é possível optar por excluir os dados ou cancelar a operação.

Uma opção mais perigosa é invocar `%db_reset` com a opção `--yes` ou `-y`, o que faz com que a exclusão seja executada sem nenhuma solicitação adicional.

Você também pode realizar a redefinição em duas etapas, assim como com a API REST:

```
%db_reset --generate-token
```

A resposta é:

```
{
  "status" : "200 OK",
  "payload" : {
    "token" : "new_token_guid"
  }
}
```

Então faça:

```
%db_reset --token new_token_guid
```

A resposta é:

```
{
```

```
  "status" : "200 OK"
}
```

Códigos de erro comuns para operações de redefinição rápida

Código de erro do Neptune	Status HTTP	Message	Exemplo
`InvalidParameterException`	400	O parâmetro de comando do sistema “*action*” tem o valor não compatível “*XXX*”.	Parâmetro inválido
`InvalidParameterException`	400	Muitos valores fornecidos para: *action*	Uma solicitação de redefinição rápida com mais de uma ação enviada com o cabeçalho “Content-type:application/x-www-form-urlencoded”.
`InvalidParameterException`	400	Campo duplicado “ação”	Uma solicitação de redefinição rápida com mais de uma ação enviada com o cabeçalho “Content-Type: application/json”.
`MethodNotAllowedException`	400	Rota inadequada: /*bad_endpoint*	Solicitação enviada a um endpoint incorreto
`MissingParameterException`	400	Parâmetros obrigatórios ausentes: [ação]	Uma solicitação de redefinição rápida não

Código de erro do Neptune	Status HTTP	Message	Exemplo
			contém o parâmetro “ação” necessário.
`ReadOnlyViolationException`	400	Não são permitidas gravações em uma instância de réplica de leitura.	Uma solicitação de redefinição rápida foi enviada a um endpoint de status ou de leitor
`AccessDeniedException`	403	Token de autenticação ausente	Uma solicitação de redefinição foi enviada sem assinaturas corretas a um endpoint de banco de dados com a IAM-Auth habilitada
`ServerShutdownException`	500	A redefinição do banco de dados está em andamento. Tente fazer a consulta novamente depois que o cluster estiver disponível.	Quando a redefinição rápida começa, ocorre uma falha nas consultas existentes e recebidas do Gremlin/Sparql.

Adicionar instâncias de leitor do Neptune a um cluster de banco de dados

Em clusters de banco de dados do Neptune, há uma instância de banco de dados principal e até 15 instâncias de leitor do Neptune. A instância de banco de dados principal oferece suporte a operações de leitura e gravação, além de realizar todas as modificações de dados no volume do cluster. Instâncias de réplica de leitor do Neptune se conectam ao mesmo volume de armazenamento da instância de banco de dados principal e só são compatíveis com operações de leitura.

Use instâncias de leitor para descarregar workloads de leitura da instância de banco de dados principal.

Recomendamos distribuir a instância principal e os leitores do Neptune no cluster de banco de dados em várias zonas de disponibilidade para melhorar a disponibilidade do cluster de banco de dados.

A seção a seguir descreve como criar uma instância de leitor no cluster de banco de dados.

Criar uma instância de leitor do Neptune usando o console

Depois de criar a instância principal para o cluster de banco de dados do Neptune, é possível adicionar outras instâncias de leitor do Neptune usando o console do Neptune.

Como criar uma instância de leitor do Neptune usando o AWS Management Console

1. Faça login no Console de Gerenciamento da AWS e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. No painel de navegação, escolha Databases (Bancos de dados).
3. Selecione o cluster de banco de dados em que deseja criar a instância de leitor.
4. Selecione Ações e, depois, Adicionar leitor.
5. Na página Criar instância de banco de dados de réplica, especifique as opções da réplica do Neptune. A tabela a seguir mostra configurações de uma réplica de leitura do Neptune.

Para esta opção...	Faça o seguinte
Classe de instância de banco de dados	Selecione uma classe de instância de banco de dados que defina os requisitos de processamento e memória para cada réplica do Neptune. Para obter uma lista atual das classes de instância de banco de dados que o Neptune oferece em diferentes regiões, consulte a página Preços do Neptune.
Availability zone	Especificar uma zona de disponibilidade. Escolha uma zona diferente da instância de banco de dados principal. A lista inclui apenas as Zonas de disponibi lidade que são mapeadas pelo grupo de sub-redes do banco de dados para o cluster do banco de dados.
Criptografia	Habilitar ou desabilitar criptografia.
Read replica source	Selecione o identificador da instância principal para criar uma réplica do Neptune.
DB instance identifier	Insira um nome para a instância exclusiva da sua conta na região selecionada. É possível optar por

Para esta opção...	Faça o seguinte
	adicionar inteligência ao nome, como incluir a zona de disponibilidade selecionada, por exemplo `neptune-us-east-1c` .
Porta de banco de dados	O número da porta na qual o banco de dados aceita conexões.
Grupo de parâmetros de banco de dados	O grupo de parâmetro dessa instância.
Exportações de log	Escolha os logs que você deseja publicar, se houver.
Atualização de versão do Auto Minor	Selecione Sim se quiser permitir que a réplica do Neptune receba automaticamente atualizações secundárias de versão de mecanismos de banco de dados do Neptune quando estiverem disponíveis. A opção Auto Minor Version Upgrade (Atualização automática de versão secundária) se aplica somente a atualizações secundárias. Ela não se aplica aos patches de manutenção do mecanismo, que são sempre aplicados automaticamente para manter a estabilidade do sistema.

6. Selecione Criar réplica de leitura para criar a instância de réplica do Neptune.

Para remover uma instância de leitor do Neptune de um cluster de banco de dados, siga as instruções em Deleting a DB instance in Amazon Neptune.

Modificar um cluster de banco de dados do Neptune usando o console

Quando você modifica uma instância de banco de dados usando o AWS Management Console, é possível aplicar as alterações na hora selecionando Apply Immediately (Aplicar imediatamente). Se você optar por aplicar as alterações imediatamente, as novas alterações e todas as alterações na fila de modificações pendentes serão aplicadas de uma vez.

Se você não optar por aplicar as alterações imediatamente, elas serão colocadas na fila de modificações pendentes. Durante a próxima janela de manutenção, todas as alterações pendentes na fila serão aplicadas.

Important

Se alguma das modificações pendentes exigir tempo de inatividade, escolher aplicar alterações imediatamente poderá causar um tempo de inatividade inesperado para a instância de banco de dados em questão. Não há tempo de inatividade para as outras instâncias de banco de dados no cluster de banco de dados.

Note

Ao modificar um cluster de banco de dados no Neptune, a configuração Aplicar imediatamente afeta somente as alterações feitas nas configurações Identificador do cluster de banco de dados e Autenticação de banco de dados do IAM. Todas as outras modificações são aplicadas imediatamente, independentemente do valor da configuração Apply Immediately (Aplicar imediatamente).

Para modificar um cluster de banco de dados usando o console

1. Faça login no Console de Gerenciamento da AWS e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. No painel de navegação, escolha Clusters e, depois, selecione o cluster de banco de dados que deseja modificar.
3. Escolha Actions (Ações) e Modify cluster (Modificar cluster). A página Modify DB cluster (Modificar cluster de banco de dados) é exibida.

4. Altere qualquer uma das configurações desejadas.

> **Note**
>
> No console, algumas alterações feitas no nível da instância só se aplicam à instância de banco de dados atual, enquanto outras se aplicam a todo o cluster de banco de dados. Para alterar uma configuração que modifica todo o cluster de banco de dados no nível de instância no console, siga as instruções em Modificar uma instância de banco de dados em um cluster de banco de dados.

5. Quando todas as alterações estiverem conforme o desejado, escolha Continue (Continuar) e verifique o resumo.
6. Para aplicar as alterações imediatamente, selecione Apply immediately (Aplicar imediatamente).
7. Na página de confirmação, revise suas alterações. Se estiverem corretas, escolha Modify cluster (Modificar cluster) para salvar as alterações.

   Para editar as alterações, selecione Back (Voltar) ou, para cancelar as alterações, escolha Cancel (Cancelar).

Modificar uma instância de banco de dados em um cluster de banco de dados

Para modificar uma instância de banco de dados em um cluster de banco de dados usando o console

1. Faça login no Console de Gerenciamento da AWS e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. No painel de navegação, selecione Instances (Instâncias) e, em seguida, selecione a instância de banco de dados a ser modificada.
3. Escolha Instance actions e, em seguida, escolha Modify. A página Modify DB Instance (Modificar instância de banco de dados) é exibida.
4. Altere qualquer uma das configurações desejadas.

> **Note**
>
> Algumas configurações se aplicam a todo o cluster de banco de dados e devem ser alteradas no nível do cluster. Para alterar essas configurações, siga as instruções em Modificar um cluster de banco de dados do Neptune usando o console.
> No AWS Management Console, algumas alterações feitas no nível da instância só se aplicam à instância de banco de dados atual, enquanto outras se aplicam a todo o cluster de banco de dados.

5. Quando todas as alterações estiverem conforme o desejado, escolha Continue (Continuar) e verifique o resumo.
6. Para aplicar as alterações imediatamente, selecione Apply immediately (Aplicar imediatamente).
7. Na página de confirmação, revise suas alterações. Se estiverem corretas, escolha Modify DB Instance (Modificar instância de banco de dados) para salvar suas alterações.

   Para editar as alterações, selecione Back (Voltar) ou, para cancelar as alterações, escolha Cancel (Cancelar).

Desempenho e escalabilidade no Amazon Neptune

Instâncias e clusters de banco de dados do Neptune são escalados em três níveis diferentes:

- Escalabilidade de armazenamento
- Escalabilidade de instâncias;
- Escalabilidade de leitura

Escalabilidade de armazenamento no Neptune

O armazenamento do Neptune é escalado automaticamente com os dados no volume do cluster. À medida que os dados aumentam, o armazenamento do volume do cluster se expande até 128 TiB em todas as regiões compatíveis, exceto na China e na GovCloud, onde é limitado a 64 TiB.

O tamanho do volume do cluster é verificado de hora em hora para determinar os custos de armazenamento.

O armazenamento consumido pelo banco de dados do Neptune é cobrado em incrementos de GB/mês, e as E/S consumidas são cobradas em incrementos de solicitação por milhão. Você paga apenas pelo armazenamento e pelas operações de E/S que o banco de dados do Neptune consome e não é necessário provisionar com antecedência.

Para obter informações sobre preços, consulte a página sobre o produto do Neptune.

Escalabilidade de instâncias no Neptune

É possível escalar o cluster de banco de dados do Neptune, conforme a necessidade, modificando a classe de instância de banco de dados para cada instância de banco de dados do cluster. O Neptune é compatível com várias classes de instâncias de banco de dados otimizadas.

Escalabilidade de leitura no Neptune

É possível obter escalabilidade de leitura para o cluster de banco de dados do Neptune criando até 15 réplicas do Neptune no cluster de banco de dados. Cada réplica do Neptune exibe os mesmos dados do volume de cluster com atraso de réplica mínimo (geralmente bem inferior a cem milissegundos após a instância principal ter gravado uma atualização). Conforme o tráfego de leitura aumenta, é possível criar mais réplicas do Neptune e conectar-se diretamente a elas para distribuir

a carga de leitura para o cluster de banco de dados. As réplicas do Neptune não precisam ser da mesma classe da instância de banco de dados que a instância principal.

Para obter informações sobre como adicionar réplicas do Neptune a um cluster de banco de dados, consulte Adicionar instâncias de leitor.

Ajuste de escala automático do número de réplicas em um cluster de banco de dados do Amazon Neptune

É possível usar o ajuste de escala automático do Neptune para ajustar automaticamente o número de réplicas do Neptune em um cluster de banco de dados para atender aos requisitos de conectividade e workload. O ajuste de escala automático permite que o cluster de banco de dados do Neptune processe aumentos na workload e, quando a workload diminui, o ajuste de escala automático remove réplicas desnecessárias para que você não pague pela capacidade não utilizada.

É possível usar o ajuste de escala automático apenas com um cluster de banco de dados do Neptune que já tenha uma instância de gravador principal e pelo menos uma instância de réplica de leitura (consulte Clusters e instâncias de banco de dados do Amazon Neptune). Além disso, todas as instâncias de réplica de leitura no cluster devem estar em um estado disponível. Se alguma réplica de leitura estiver em um estado diferente de disponível, o ajuste de escala automático do Neptune não fará nada até que todas as réplicas de leitura no cluster estejam disponíveis.

Consulte Criar um cluster de banco de dados se você precisar criar um cluster.

Usando o AWS CLI, você define e aplica uma política de escalabilidade ao cluster de banco de dados. Você também pode usar o AWS CLI para editar ou excluir sua política de auto-scaling. A política especifica os seguintes parâmetros de ajuste de escala automático:

- O número mínimo e máximo de réplicas a serem mantidas no cluster.
- Um `ScaleOutCooldown` intervalo entre a atividade de escalabilidade de adição de réplicas e um `ScaleInCooldown` intervalo entre a atividade de escalabilidade de exclusão de réplicas.
- A CloudWatch métrica e o valor do gatilho métrico para aumentar ou diminuir a escala.

A frequência das ações de ajuste de escala automático do Neptune é reduzida de várias maneiras:

- Inicialmente, para que o ajuste de escala automático adicione ou exclua um leitor, o alarme alto `CPUUtilization` deve ser violado por pelo menos três minutos ou o alarme baixo deve ser violado por pelo menos 15 minutos.
- Após a primeira adição ou exclusão, a frequência das ações subsequentes de ajuste de escala automático do Neptune é limitada pelas configurações `ScaleOutCooldown` e `ScaleInCooldown` na política de ajuste de escala automático.

Se a CloudWatch métrica que você está usando atingir o limite alto especificado em sua política e se o `ScaleOutCooldown` intervalo tiver decorrido desde a última ação de auto-scaling, e se seu cluster de banco de dados ainda não tiver o número máximo de réplicas que você definiu, o auto-scaling do Neptune criará uma nova réplica usando o mesmo tipo de instância da instância primária do cluster de banco de dados.

Da mesma forma, se a métrica atingir o limite baixo especificado e se o intervalo `ScaleInCooldown` tiver decorrido desde a última ação de ajuste de escala automático, e se o cluster de banco de dados tiver mais do que o número mínimo de réplicas especificado, o ajuste de escala automático do Neptune excluirá uma das réplicas.

> **Note**
>
> O ajuste de escala automático do Neptune remove somente as réplicas criadas. Ele não remove réplicas preexistentes.

Usando o parâmetro de cluster de banco de dados neptune_autoscaling_config, também é possível especificar o tipo de instância das novas réplicas de leitura criadas pelo ajuste de escala automático do Neptune, as janelas de manutenção dessas réplicas de leitura e as tags a serem associadas a cada uma das novas réplicas de leitura. Você fornece essas configurações em uma string JSON como o valor do parâmetro `neptune_autoscaling_config`, desta forma:

```
"{
  \"tags\": [
    { \"key\" : \"reader tag-0 key\", \"value\" : \"reader tag-0 value\" },
    { \"key\" : \"reader tag-1 key\", \"value\" : \"reader tag-1 value\" },
  ],
  \"maintenanceWindow\" : \"wed:12:03-wed:12:33\",
  \"dbInstanceClass\" : \"db.r5.xlarge\"
}"
```

Observe que as aspas dentro da string JSON devem ter todas um caractere de escape que é uma barra invertida (\). Todos os espaços em branco na string são opcionais, como de costume.

Qualquer uma das três definições de configuração não especificadas no parâmetro `neptune_autoscaling_config` é copiada da configuração da instância de gravador principal do cluster de banco de dados.

Quando o ajuste de escala automático adiciona uma nova instância de réplica de leitura, ele inclui no ID da instância de banco de dados o prefixo `autoscaled-reader` (por exemplo, `autoscaled-reader-7r7t7z3lbd-20210828`). Ele também adiciona uma tag a cada réplica de leitura criada com a chave `autoscaled-reader` e um valor de `TRUE`. É possível visualizar essa tag na guia Tags da página de detalhes da instância de banco de dados no AWS Management Console.

```
"key" : "autoscaled-reader",  "value" : "TRUE"
```

O nível de promoção de todas as instâncias de réplica de leitura criadas pelo ajuste de escala automático é o de menor prioridade, que é 15 por padrão. Isso significa que, durante um failover, qualquer réplica com uma prioridade maior, como uma criada manualmente, será promovida primeiro. Consulte Tolerância a falhas para um cluster de banco de dados do Neptune.

O auto-scaling do Neptune é implementado usando o Application Auto Scaling com uma política de escalabilidade de rastreamento de metas que usa uma métrica do Neptune como uma métrica predefinida. `CPUUtilization` CloudWatch

Usar o ajuste de escala automático em um cluster de banco de dados sem servidor do Neptune

O Neptune Serverless responde com uma velocidade muito maior do que o ajuste de escala automático do Neptune quando a demanda excede a capacidade de uma instância e aumenta a escala da instância verticalmente em vez de adicionar outra instância. Enquanto o ajuste de escala automático é projetado para corresponder a aumentos ou diminuições relativamente estáveis na workload, a tecnologia sem servidor é excelente para lidar com picos rápidos e oscilações na demanda.

Compreendendo os pontos fortes, você pode combinar o ajuste de escala automático e a tecnologia sem servidor para criar uma infraestrutura flexível que lidará com as mudanças na workload com eficiência e atenderá à demanda minimizando os custos.

Para permitir que o ajuste de escala automático funcione com eficiência com a tecnologia sem servidor, é importante definir a configuração do `maxNCU` do cluster sem servidor alta o suficiente para acomodar picos e breves mudanças na demanda. Caso contrário, alterações transitórias não acionam a escalabilidade sem servidor, o que pode fazer com que o ajuste de escala automático gere muitas instâncias adicionais desnecessárias. Se o `maxNCU` estiver definido como alto o suficiente, o ajuste de escala automático sem servidor pode lidar com essas alterações de forma mais rápida e econômica.

Como habilitar o ajuste de escala automático do Amazon Neptune

O ajuste de escala automático só pode ser habilitado para um cluster de banco de dados do Neptune usando a AWS CLI. Não é possível habilitar o ajuste de escala automático usando o. AWS Management Console.

Além disso, o ajuste de escala automático não é compatível com as seguintes regiões da Amazon:

- África (Cidade do Cabo): `af-south-1`
- Oriente Médio (Emirados Árabes Unidos): `me-central-1`
- AWS GovCloud (Leste dos EUA): `us-gov-east-1`
- AWS GovCloud (Oeste dos EUA): `us-gov-west-1`

Habilitar o ajuste de escala automático para um cluster de banco de dados do Neptune envolve três etapas:

1. Registrar o cluster de banco de dados com o Application Auto Scaling

A primeira etapa para habilitar o ajuste de escala automático para um cluster de banco de dados do Neptune é registrar o cluster no Application Auto Scaling usando a AWS CLI ou um dos SDKs do Application Auto Scaling. O cluster já deve ter uma instância principal e pelo menos uma instância de réplica de leitura:

Por exemplo, para registrar um cluster para ser escalado automaticamente com de uma a oito réplicas adicionais, você pode usar o AWS CLI `register-scalable-target`comando da seguinte forma:

```
aws application-autoscaling register-scalable-target \
  --service-namespace neptune \
  --resource-id cluster:(your DB cluster name) \
  --scalable-dimension neptune:cluster:ReadReplicaCount \
  --min-capacity 1 \
  --max-capacity 8
```

Isso equivale a usar a operação da API `RegisterScalableTarget` do Application Auto Scaling.

O comando AWS CLI `register-scalable-target` usa os seguintes parâmetros:

- **`service-namespace`**: defina como `neptune`.

Esse parâmetro equivale ao parâmetro `ServiceNamespace` na API do Application Auto Scaling.

- **`resource-id`**: defina como o identificador de recurso do cluster de banco de dados do Neptune. O tipo de recurso é `cluster`, seguido por dois pontos (“`:`“) e, depois, pelo nome do cluster de banco de dados.

  Esse parâmetro equivale ao parâmetro `ResourceID` na API do Application Auto Scaling.
- **`scalable-dimension`**: a dimensão escalável nesse caso é o número de instâncias de réplica no cluster de banco de dados, então você define esse parâmetro como `neptune:cluster:ReadReplicaCount`.

  Esse parâmetro equivale ao parâmetro `ScalableDimension` na API do Application Auto Scaling.
- **`min-capacity`**: o número mínimo de instâncias de réplica de banco de dados de leitor a serem gerenciadas pelo Application Auto Scaling. Esse valor deve ser definido no intervalo de 0 a 15 e deve ser igual ou menor que o valor especificado para o número máximo de réplicas do Neptune em `max-capacity`. Deve haver pelo menos um leitor no cluster de banco de dados para que o ajuste de escala automático funcione.

  Esse parâmetro equivale ao parâmetro `MinCapacity` na API do Application Auto Scaling.
- **`max-capacity`**: o número máximo de instâncias de réplicas de banco de dados de leitor no cluster de banco de dados, incluindo instâncias preexistentes e novas instâncias gerenciadas pelo Application Auto Scaling. Esse valor deve ser definido entre 0 e 15 e deve ser igual ou maior que o valor especificado para o número mínimo de réplicas do Neptune em `min-capacity`.

  O `max-capacity` AWS CLI parâmetro é equivalente ao `MaxCapacity` parâmetro na API Application Auto Scaling.

Quando você registra o cluster de banco de dados, o Application Auto Scaling cria um perfil vinculado ao serviço `AWSServiceRoleForApplicationAutoScaling_NeptuneCluster`. Para obter mais informações, consulte Service-linked roles for Application auto-scaling, no Guia do usuário do Application Auto Scaling.

2. Definir uma política de ajuste de escala automático a ser usada com o cluster de banco de dados

Uma política de escalabilidade de rastreamento de destino é definida como um objeto de texto JSON que também pode ser salvo em um arquivo de texto. Para Neptune, essa política atualmente só pode

usar a métrica `CPUUtilization` CloudWatch Neptune como uma métrica predefinida chamada. `NeptuneReaderAverageCPUUtilization`

Veja um exemplo de política de configuração de escalabilidade de rastreamento de destino para o Neptune:

```
{
  "PredefinedMetricSpecification": { "PredefinedMetricType":
 "NeptuneReaderAverageCPUUtilization" },
  "TargetValue": 60.0,
  "ScaleOutCooldown" : 600,
  "ScaleInCooldown" : 600
}
```

O elemento **`TargetValue`** aqui contém a porcentagem de utilização da CPU acima da qual o ajuste de escala automático se expande (ou seja, adiciona mais réplicas) e abaixo da qual ele é reduzido (ou seja, exclui réplicas). Nesse caso, a porcentagem de destino que aciona a escalabilidade é `60.0%`.

O elemento **`ScaleInCooldown`** especifica a quantidade de tempo, em segundos, após a conclusão de redução da escala antes que outra redução possa ser iniciada. O padrão é trezentos segundos. Aqui, o valor de seiscentos especifica que devem decorrer pelo menos dez minutos entre a conclusão de uma exclusão de réplica e o início de outra.

O elemento **`ScaleOutCooldown`** especifica a quantidade de tempo, em segundos, após a conclusão de expansão da escala antes que outra expansão possa ser iniciada. O padrão é trezentos segundos. Aqui, o valor de seiscentos especifica que devem decorrer pelo menos dez minutos entre a conclusão de uma adição de réplica e o início de outra.

O elemento **`DisableScaleIn`** é um valor booliano que, se estiver presente e definido como `true` desabilitará totalmente a redução da escala, o que significa que o ajuste de escala automático poderá adicionar réplicas, mas nunca removerá nenhuma. Por padrão, a redução da escala está habilitada e `DisableScaleIn` é `false`.

Depois de registrar o cluster de banco de dados do Neptune com o Application Auto Scaling e definir uma política de escalabilidade JSON em um arquivo de texto, aplique a política de escalabilidade ao cluster de banco de dados registrado. Você pode usar o AWS CLI `put-scaling-policy`comando para fazer isso, com parâmetros como os seguintes:

```
aws application-autoscaling put-scaling-policy \
```

```
  --policy-name (name of the scaling policy) \
  --policy-type TargetTrackingScaling \
  --resource-id cluster:(name of your Neptune DB cluster) \
  --service-namespace neptune \
  --scalable-dimension neptune:cluster:ReadReplicaCount \
  --target-tracking-scaling-policy-configuration file://(path to the JSON configuration
file)
```

Quando você aplica a política de ajuste de escala automático, o ajuste de escala automático é habilitado no cluster de banco de dados.

Você também pode usar o AWS CLI `put-scaling-policy`comando para atualizar uma política de auto-scaling existente.

Consulte também PutScalingPolítica na Referência da API Application Auto Scaling.

Remover o ajuste de escala automático de um cluster de banco de dados do Neptune

Para remover o auto-scaling de um cluster de banco de dados Neptune, use os comandos delete-scaling-policy e deregister-scalable-target. AWS CLI

Manter o cluster de banco de dados do Amazon Neptune

O Neptune realiza manutenção periódica em todos os recursos que usa, incluindo:

- Substituir o hardware subjacente conforme necessário. Isso acontece em segundo plano, sem que você precise realizar nenhuma ação e geralmente não afeta as operações.
- Atualizar o sistema operacional subjacente. As atualizações do sistema operacional das instâncias no cluster de banco de dados são realizadas para melhorar o desempenho e a segurança. Portanto, você deve concluí-las o mais rápido possível. Em geral, essas atualizações demoram cerca de dez minutos. As atualizações do sistema operacional não alteram a versão do mecanismo de banco de dados nem a classe de uma instância de banco de dados.

  Em geral, é melhor atualizar primeiro as instâncias do leitor em um cluster de banco de dados e depois a instância do gravador. Atualizar os leitores e o gravador ao mesmo tempo pode causar um período de inatividade no caso de um failover. O backup das instâncias de banco de dados não é feito automaticamente antes de uma atualização do sistema operacional. Dessa forma, faça backups manuais antes de aplicar uma atualização do sistema operacional.
- Atualizar o mecanismo do banco de dados Neptune. O Neptune lança regularmente uma variedade de atualizações do mecanismo para introduzir novos recursos e melhorias e corrigir bugs.

Números das versões do mecanismo

Numeração de versão antes da versão 1.3.0.0 do mecanismo

Antes de novembro de 2019, o Neptune era compatível apenas com uma versão do mecanismo por vez, e os números de versão do mecanismo tinham todos o mesmo formato, `1.0.1.0.200`*`<xxx>`*, em que `xxx` era o número do patch. As novas versões do mecanismo foram todas lançadas como patches para versões anteriores.

Em novembro de 2019, o Neptune passou a ser compatível com várias versões, oferecendo aos clientes um melhor controle dos caminhos de atualização. Consequentemente, a numeração de versões do mecanismo foi alterada.

De novembro de 2019 até a versão 1.3.0.0 do mecanismo, os números de versão tinham cinco partes. Por exemplo, no número de versão `1.0.2.0.R2`:

- A primeira parte sempre foi 1.

- A segunda parte, (`0` em `1.0.2.0.R2`), era o número da versão principal do banco de dados.
- A terceira e a quarta parte, (`2.0` em `1.0.2.0.R2`), eram números de versão secundária.
- A quinta parte, (`R2` em `1.0.2.0.R2`), era o número do patch.

A maioria das atualizações era atualizações de patches, e a distinção entre patches e atualizações de versões secundárias nem sempre era clara.

Numeração de versão a partir da versão 1.3.0.0 do mecanismo

A partir da versão 1.3.0.0 do mecanismo, o Neptune mudou a forma como as atualizações do mecanismo são numeradas e gerenciadas.

Os números de versão do mecanismo agora têm quatro partes, cada uma correspondendo a um tipo de versão, da seguinte forma:

product-version.*major-version*.*minor-version*.*patch-version*

Alterações ininterruptas do tipo que foram lançadas anteriormente como patches agora são lançadas como versões secundárias que você pode gerenciar usando a configuração da instância `AutoMinorVersionUpgrade`.

Isso significa que, se você quiser, poderá receber uma notificação sempre que uma nova versão secundária for lançada, basta assinar o evento `RDS-EVENT-0156` (consulte Assinar a notificação de eventos do Neptune).

As versões de patches agora estão reservadas para correções direcionadas urgentes e são numeradas usando a última parte do número da versão (`*.*.*.1`, `*.*.*.2` e assim por diante).

Diferentes tipos de versões do mecanismo no Amazon Neptune

Os quatro tipos de versão do mecanismo que correspondem às quatro partes de um número de versão do mecanismo são os seguintes:

- Versão do produto: é alterada somente se o produto passar por mudanças amplas e fundamentais na funcionalidade ou interface. A versão atual do produto Neptune é 1.
- Versão principal: a versões principais introduzem novos recursos importantes e mudanças significativas e geralmente têm uma vida útil de pelo menos dois anos.
- Versão secundária: as versões secundárias podem conter novos recursos, melhorias e correções de erros, mas não contêm alterações significativas. Você pode escolher se deseja ou não que

sejam aplicadas automaticamente durante a próxima janela de manutenção, além de optar por receber uma notificação sempre que for lançada.

- Versão de patch: as versões de patch são lançadas somente para tratar de correções de erros urgentes ou atualizações críticas de segurança. Raramente contêm alterações significativas e são aplicadas automaticamente durante a próxima janela de manutenção após o lançamento.

Atualizações da versão principal do Amazon Neptune

Uma atualização de versão principal geralmente introduz um ou mais recursos novos e importantes e geralmente contém alterações significativas. Normalmente, tem uma vida útil de suporte de cerca de dois anos. As versões principais do Neptune estão listadas nas versões do mecanismo, junto com a data em que foram lançadas e o fim da vida útil estimado.

As atualizações da versão principal são totalmente opcionais até que a versão principal utilizada chegue ao fim da vida útil. Se você optar por atualizar para uma nova versão principal, você deverá instalar a nova versão usando o AWS CLI ou o console o Neptune, conforme descrito em Atualizações de versão principal.

No entanto, se a versão principal utilizada chegar ao fim da vida útil, você receberá uma notificação de que é necessário atualizar para uma versão principal mais recente. Dessa forma, se você não fizer a atualização dentro de um período de carência após a notificação, uma atualização para a versão principal mais recente será programada automaticamente para ocorrer durante a próxima janela de manutenção. Consulte Vida útil da versão do mecanismo para obter mais informações.

Atualizações da versão secundária do Amazon Neptune

A maioria das atualizações do mecanismo do Neptune são atualizações de versões secundárias. Elas acontecem com bastante frequência e não contêm alterações significativas.

Se o campo `AutoMinorVersionUpgrade` estiver definido como `true` na instância (primária) do gravador do cluster de banco de dados, as atualizações da versão secundária serão aplicadas automaticamente a todas as instâncias no cluster de banco de dados durante a próxima janela de manutenção após o lançamento.

Se o campo `AutoMinorVersionUpgrade` estiver definido como `false` na instância do gravador do cluster de banco de dados, elas serão aplicadas somente se você instalá-las explicitamente.

> **Note**
>
> As atualizações de versões secundárias são independentes (não dependem das atualizações anteriores da versão secundária para a mesma versão principal) e cumulativas (elas contêm todos os recursos e correções introduzidos nas atualizações de versões secundárias anteriores). Isso significa que você pode instalar qualquer atualização de versão secundária, independentemente de ter instalado ou não as anteriores.

É fácil acompanhar os lançamentos de versões secundárias assinando o evento `RDS-EVENT-0156` (consulte Assinar a notificação de eventos do Neptune). Em seguida, você receberá notificações sempre que uma nova versão secundária for lançada.

Além disso, independentemente de você assinar ou não as notificações, sempre poderá verificar quais atualizações estão pendentes.

Atualizações de versão de patch do Amazon Neptune

No caso de problemas de segurança ou outros defeitos graves que afetam a confiabilidade da instância, o Neptune implanta patches obrigatórios. Eles são aplicados a todas as instâncias no cluster de banco de dados durante próxima janela de manutenção, sem qualquer intervenção de sua parte.

Uma versão de patch é implantada somente quando os riscos de não implantá-la superam os riscos e o tempo de inatividade associados à implantação. As versões de patch não ocorrem com frequência (geralmente uma vez a cada poucos meses) e raramente exigem mais do que uma fração de sua janela de manutenção para serem aplicadas.

Planejar a vida útil da versão principal do mecanismo do Amazon Neptune

As versões do mecanismo do Neptune quase sempre chegam ao fim da vida útil no final de um trimestre civil. As exceções ocorrem somente quando surgem problemas importantes de segurança ou disponibilidade.

Quando uma versão do mecanismo chegar ao fim da vida útil, você precisará atualizar o banco de dados Neptune para uma versão mais recente.

Em geral, as versões do mecanismo do Neptune continuam disponíveis da seguinte forma:

- Versões secundárias do mecanismo: as versões secundárias do mecanismo permanecem disponíveis por pelo menos seis meses após o lançamento.
- Versões principais do mecanismo: as versões principais do mecanismo permanecem disponíveis por pelo menos 12 meses após o lançamento.

Pelo menos três meses antes de uma versão do mecanismo chegar ao fim da vida útil, AWS enviará uma notificação automática ao endereço de e-mail associado à conta da AWS e publicará a mesma mensagem no AWS Health Dashboard. Isso dará tempo para planejar e se preparar para a atualização.

Quando uma versão do mecanismo chegar ao fim da vida útil, você não poderá mais criar clusters ou instâncias usando essa versão, nem o ajuste de escala automático poderá criar instâncias usando essa versão.

Uma versão do mecanismo que realmente chegue ao fim da vida útil será atualizada automaticamente durante uma janela de manutenção. A mensagem enviada a você três meses antes do fim da vida útil da versão do mecanismo conterá detalhes sobre o que essa atualização automática envolverá, incluindo a versão para a qual o mecanismo será atualizado automaticamente, o impacto nos clusters de banco de dados e as ações que recomendamos.

Important

Você é responsável por manter as versões do mecanismo de banco de dados atualizadas. A AWS incentiva todos os clientes a atualizar os bancos de dados para a versão mais recente do mecanismo, a fim de se beneficiarem das proteções de segurança, privacidade e disponibilidade mais atualizadas. Se você operar o banco de dados em um mecanismo ou um software não compatível após a data de defasagem (“Mecanismo herdado”), enfrentará uma maior probabilidade de riscos operacionais, segurança e privacidade, incluindo eventos de inatividade.

A operação do banco de dados em qualquer mecanismo está sujeita ao Contrato que rege o uso dos Serviços da AWS. Os Mecanismos herdados não estão disponíveis ao público em geral. A AWS não oferece mais suporte ao Mecanismo herdado e a AWS pode limitar o acesso ou o uso de qualquer Mecanismo herdado a qualquer momento, se a AWS determinar que o Mecanismo herdado representa um risco de segurança ou responsabilidade, ou um risco de danos, aos Serviços da AWS, a suas Afiliadas ou a terceiros. Sua decisão de continuar executando Seu conteúdo em um Mecanismo herdado pode fazer com que Seu conteúdo fique indisponível, corrompido ou irrecuperável. Os

bancos de dados executados em um Mecanismo herdado estão sujeitos às exceções do Acordo de Serviço (SLA).
BANCOS DE DADOS E SOFTWARES RELACIONADOS EXECUTADOS EM UM MECANISMO HERDADO CONTÊM BUGS, ERROS, DEFEITOS E/OU COMPONENTES NOCIVOS. CONSEQUENTEMENTE, E NÃO OBSTANTE QUALQUER DISPOSIÇÃO EM CONTRÁRIO NO CONTRATO OU NOS TERMOS DO SERVIÇO, A AWS ESTÁ FORNECENDO O MECANISMO HERDADO “NO ESTADO EM QUE SE ENCONTRA”.

Gerenciar atualizações do mecanismo no cluster de banco de dados do Neptune

Note

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados em todas essas instâncias, ocorrerá um tempo de inatividade de vinte ou trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados. Em raras ocasiões, um failover Multi-AZ pode ser necessário para concluir uma atualização de manutenção em uma instância.
Para atualizações de versões principais que podem levar mais tempo para serem aplicadas, você pode usar uma estratégia de implantação azul/verde para minimizar o tempo de inatividade.

Determinar qual versão do mecanismo você está usando no momento

Você pode usar o comando `get-engine-status` do AWS CLI para verificar qual versão do mecanismo que o cluster de banco de dados está usando atualmente:

```
aws neptunedata get-engine-status
```

A saída JSON inclui um campo `"dbEngineVersion"` como este:

```
  "dbEngineVersion": "1.3.0.0",
```

Verifique quais atualizações estão pendentes e disponíveis

Você pode verificar as atualizações pendentes do cluster de banco de dados do usando o console do Neptune. Selecione Bancos de dados na coluna esquerda e, em seguida, o cluster de banco de dados no painel de bancos de dados. As atualizações pendentes estão listadas na coluna Manutenção. Se você selecionar Ações e, em seguida, Manutenção, você tem três opções:

- Atualizar agora.
- Atualizar na próxima janela.
- Adiar atualização.

Você pode listar as atualizações pendentes do mecanismo usando a AWS CLI da seguinte forma:

```
aws neptune describe-pending-maintenance-actions \
  --resource-identifier (ARN of your DB cluster)
  --region (your region) \
  --engine neptune
```

Também é possível listar as versões do mecanismo disponíveis usando a AWS CLI da seguinte forma:

```
aws neptune describe-db-engine-versions \
  --region (your region) \
  --engine neptune
```

A lista de versões disponíveis do mecanismo inclui somente as versões com um número maior do que a atual e para as quais há um caminho de atualização definido.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Uma atualização secundária pode introduzir novos recursos ou comportamentos que afetam o código mesmo sem nenhuma alteração significativa.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é usar a solução de implantação azul/verde do Neptune. Dessa forma, você pode executar aplicações e consultas na nova versão sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

Janela de manutenção do Neptune

A janela de manutenção semanal é um período de 30 minutos durante o qual as atualizações programadas do mecanismo e outras alterações do sistema são aplicadas. A maioria dos eventos de manutenção é concluída durante a janela de 30 minutos, embora os eventos de manutenção mais longos possam levar mais tempo para serem concluídos.

Cada cluster de banco de dados tem uma janela de manutenção semanal de 30 minutos. Se você não especificar um horário de preferência ao criar o cluster de banco de dados, o Neptune selecionará aleatoriamente um dia da semana e também atribuirá aleatoriamente um período de 30 minutos a partir de um bloco de 8 horas que varia de acordo com a região.

Aqui, por exemplo, estão os blocos de tempo de 8 horas para janelas de manutenção usadas em várias regiões da AWS:

Região	Bloco de hora
Região Oeste dos EUA (Oregon)	6h às 14h (UTC)

Região	Bloco de hora
Região Oeste dos EUA (Norte da Califórnia).	6h às 14h (UTC)
Região Leste dos EUA (Ohio)	3h às 11h (UTC)
Região Europa (Irlanda)	22h às 6h (UTC)

A janela de manutenção determina quando as operações pendentes começam, e a maioria das operações de manutenção é concluída dentro da janela, mas tarefas de manutenção maiores podem continuar além do horário de término da janela.

Mover a janela de manutenção do cluster de banco de dados

O ideal é que sua janela de manutenção caia no momento em que o cluster estiver em menor uso. Se isso não se aplica à sua janela atual, você pode movê-la para um horário melhor, da seguinte forma:

Para alterar a janela de manutenção do cluster de banco de dados

1. Faça login no Console de Gerenciamento da AWS e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. No painel de navegação, escolha Bancos de dados.
3. Escolha o cluster de banco de dados para o qual deseja alterar a janela de manutenção.
4. Selecione Modify.
5. Escolha Mostrar mais na parte inferior da página Modificar cluster.
6. Na seção Janela de manutenção preferencial, defina o dia, a hora e a duração da janela de manutenção conforme sua preferência.
7. Escolha Next (Próximo).

   Na página de confirmação, revise suas alterações.
8. Para aplicar as alterações à janela de manutenção imediatamente, selecione Apply immediately (Aplicar imediatamente).
9. Escolha Enviar para aplicar as alterações.

Para editar as alterações, selecione Anterior ou, para cancelar as alterações, escolha Cancelar.

Usar **`AutoMinorVersionUpgrade`** para controlar atualizações automáticas de versões secundárias

> **Important**
>
> `AutoMinorVersionUpgrade` só é eficaz para atualizações de versões secundárias posterior à versão 1.3.0.0 do mecanismo.

Se o campo `AutoMinorVersionUpgrade` estiver definido como `true` na instância (primária) do gravador do cluster de banco de dados, as atualizações da versão secundária serão aplicadas automaticamente a todas as instâncias no cluster de banco de dados durante a próxima janela de manutenção após o lançamento.

Se o campo `AutoMinorVersionUpgrade` estiver definido como `false` na instância do gravador do cluster de banco de dados, elas serão aplicadas somente se você instalá-las explicitamente.

> **Note**
>
> As versões de patch (`*.*.*.1`, `*.*.*.2` etc.) são sempre instaladas automaticamente durante a próxima janela de manutenção, independentemente de como o parâmetro `AutoMinorVersionUpgrade` está definido.

Você pode definir `AutoMinorVersionUpgrade` usando o AWS Management Console da seguinte forma:

Para definir **`AutoMinorVersionUpgrade`** usando o console do Neptune

1. Faça login no Console de Gerenciamento da AWS e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. No painel de navegação, escolha Databases (Bancos de dados).
3. Escolha a instância primária (gravador) do cluster de banco de dados para o qual você deseja definir `AutoMinorVersionUpgrade`.
4. Selecione Modify.

5. Escolha Mostrar mais na parte inferior da página Modificar cluster.
6. Na parte inferior da página expandida, escolha Ativar atualização automática de versão secundária ou Desabilitar o upgrade automático da versão secundária.
7. Escolha Next (Próximo).

   Na página de confirmação, revise suas alterações.
8. Para aplicar as alterações na atualização automática de versões secundárias, selecione Aplicar imediatamente.
9. Escolha Enviar para aplicar as alterações.

   Para editar as alterações, selecione Anterior ou, para cancelar as alterações, escolha Cancelar.

Você também pode usar a AWS CLI para definir o campo `AutoMinorVersionUpgrade`. Por exemplo, para defini-lo como `true`, você pode usar um comando como este:

```
aws neptune modify-db-instance \
  --db-instance-identifier (the ID of your cluster's writer instance) \
  --auto-minor-version-upgrade \
  --apply-immediately
```

Da mesma forma, para defini-lo como `false`, use um comando como este:

```
aws neptune modify-db-instance \
  --db-instance-identifier (the ID of your cluster's writer instance) \
  --no-auto-minor-version-upgrade \
  --apply-immediately
```

Instalar atualizações no mecanismo do Neptune manualmente

Como instalar uma atualização da versão principal do mecanismo

As versões principais do mecanismo sempre devem ser instaladas manualmente. Para minimizar o tempo de inatividade e fornecer bastante tempo para testes e validação, a melhor maneira de instalar uma nova versão principal geralmente é usar a solução de implantação azul/verde do Neptune.

Em alguns casos, você também pode usar o modelo AWS CloudFormation com o qual criou seu cluster de banco de dados para instalar uma atualização da versão principal (consulte Usando um

AWS CloudFormation modelo para atualizar a versão do mecanismo do seu cluster de banco de dados Neptune).

Se você quiser instalar uma atualização da versão principal imediatamente, você pode usar um comando da CLI como o seguinte:

```
aws neptune modify-db-cluster \
  --db-cluster-identifier (identifier for your neptune cluster) \
  --engine neptune \
  --engine-version (the new engine version) \
  --apply-immediately
```

Especifique a versão do mecanismo para a qual você deseja atualizar. Caso contrário, o mecanismo poderá ser atualizado para uma versão que não seja a mais recente nem a esperada.

Em vez de `--apply-immediately`, é possível especificar `--no-apply-immediately`.

Se o cluster usar um grupo de parâmetros de cluster personalizado, use este parâmetro para especificá-lo:

```
  --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Da mesma forma, se alguma instância no cluster usar um grupo de parâmetros de banco de dados personalizado, use este parâmetro para especificá-lo:

```
  ---db-instance-parameter-group-name (name of the custom instance parameter group)
```

Instalar uma atualização da versão secundária do mecanismo usando o AWS Management Console

Para fazer uma atualização da versão secundária usando o console Neptune

1. Faça login no Console de Gerenciamento da AWS e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. No painel de navegação, escolha Bancos de dados e selecione o cluster de banco de dados que você deseja modificar.
3. Selecione Modify.
4. Em Especificações da instância, escolha a nova versão para a qual você deseja atualizar.
5. Escolha Next (Próximo).

6. Para aplicar alterações imediatamente, escolha Aplicar imediatamente.
7. Escolha Enviar para atualizar o cluster de banco de dados.

Instalar uma atualização da versão secundária do mecanismo usando o AWS CLI

Você pode usar um comando como o seguinte para realizar uma atualização da versão secundária sem esperar pela próxima janela de manutenção:

```
aws neptune modify-db-cluster \
  --db-cluster-identifier (your-neptune-cluster) \
  --engine-version (new-engine-version) \
  --apply-immediately
```

Se você estiver fazendo a atualização manual com a AWS CLI, inclua a versão do mecanismo para a qual deseja atualizar. Caso contrário, o mecanismo poderá ser atualizado para uma versão que não seja a mais recente nem a esperada.

Realizar a atualização para a versão 1.2.0.0 ou posterior do mecanismo a partir de uma versão anterior à 1.2.0.0

A versão 1.2.0.0 do mecanismo introduziu algumas alterações significativas que podem tornar a atualização de uma versão anterior mais complicada do que o normal:

- A versão 1.2.0.0 do mecanismo introduziu um novo formato para grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados. Como resultado, se você estiver atualizando de uma versão de mecanismo anterior à 1.2.0.0 para a versão 1.2.0.0 ou posterior, deverá recriar todos os grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados existentes usando a família de grupos de parâmetros `neptune1.2`. As versões anteriores usavam a família de grupos de parâmetros `neptune1`, e esses grupos de parâmetros não funcionarão com a versão 1.2.0.0 e posterior. Consulte Grupos de parâmetros do Amazon Neptune para obter mais informações.
- A versão 1.2.0.0 do mecanismo também introduziu um novo formato para undo logs. Como resultado, todos os undo logs criados por uma versão anterior do mecanismo devem ser eliminados e a métrica `UndoLogsListSize` do CloudWatch deve cair para zero para que seja possível iniciar qualquer atualização de uma versão anterior à 1.2.0.0. Se houver muitos registros de undo logs (duzentos mil ou mais) ao tentar iniciar uma atualização, a tentativa de atualização poderá expirar enquanto aguarda a conclusão da limpeza dos undo logs.

É possível acelerar a taxa de limpeza atualizando a instância de gravador do cluster, que é onde a limpeza ocorre. Fazer isso antes de tentar realizar a atualização pode reduzir o número de undo logs antes de começar. Aumentar o tamanho do gravador para um tipo de instância 24XL pode aumentar a taxa de limpeza para mais de um milhão de registros por hora.

Se a métrica `UndoLogsListSize` do CloudWatch for extremamente grande, abrir um caso de suporte pode ajudar a examinar estratégias adicionais para reduzi-la.

- Por fim, houve uma alteração significativa na versão 1.2.0.0. Ela afeta o código anterior que usava o protocolo Bolt com autenticação do IAM. A partir da versão 1.2.0.0, o Bolt precisa de um caminho de recursos para a assinatura do IAM. Em Java, a definição do caminho de recursos pode ser assim: `request.setResourcePath("/openCypher"));`. Em outras linguagens, o `/openCypher` pode ser anexado ao URI do endpoint. Consulte Usar o protocolo Bolt para ver exemplos.

Usando um AWS CloudFormation modelo para atualizar a versão do mecanismo do seu cluster de banco de dados Neptune

Você pode reutilizar o modelo do AWS CloudFormation Neptune que você usou para criar seu Neptune DB Cluster para atualizar sua versão do mecanismo.

As atualizações da versão do mecanismo do Neptune podem ser secundárias ou principais. Usar um AWS CloudFormation modelo pode ajudar nas principais atualizações de versões, que geralmente contêm alterações significativas. Como as atualizações de versões principais podem conter as alterações de banco de dados incompatíveis com as aplicações existentes, talvez você também precise fazer as alterações nas aplicações durante a atualização. Sempre teste antes de realizar a atualização e é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados.

Observe que você precisa fazer uma atualização de mecanismo separada para cada versão principal. Você não pode ignorar uma versão principal e realizar a atualização diretamente para a versão principal a seguir.

Antes de 17 de maio de 2023, se você usava a pilha AWS CloudFormation Neptune para atualizar sua versão do mecanismo, ela simplesmente criava um novo cluster de banco de dados vazio no lugar do seu atual. A partir de 17 de maio de 2023, no entanto, a pilha AWS CloudFormation Neptune agora oferece suporte a atualizações de mecanismos locais que preservam seus dados existentes.

Para uma atualização de versão principal, o modelo deve definir as seguintes propriedades em `DBCluster`:

- `DBClusterParameterGroup` (personalizado ou padrão)
- `DBInstanceParameterGroupName`
- `EngineVersion`

Da mesma forma, para DBInstances anexadas ao DBCluster, você deve definir:

- `DBParameterGroup` (personalizado/padrão)

Garanta que todos os grupos de parâmetros estejam definidos no modelo, sejam eles padrão ou personalizados.

No caso de um grupo de parâmetros personalizado, garanta que a família do grupo de parâmetros personalizado existente seja compatível com a nova versão do mecanismo. As versões do mecanismo anteriores à 1.2.0.0 usavam a família de grupos de parâmetros `neptune1`, enquanto as versões do mecanismo a partir da 1.2.0.0 exigem a família de grupos de parâmetros `neptune1.2`. Consulte Grupos de parâmetros do Amazon Neptune para obter mais informações.

Para atualizações de versões principais do mecanismo, especifique um grupo de parâmetros com a família apropriada no campo `DBInstanceParameterGroupName` do `DBCluster`.

Um grupo de parâmetros padrão deve ser atualizado para um que seja compatível com a nova versão do mecanismo.

Observe que o Neptune reinicia automaticamente as instâncias de banco de dados após uma atualização do mecanismo.

Tópicos

Exemplo: atualização secundária do mecanismo de 1.2.0.1 para 1.2.0.2

Encontre o cluster de banco de dados que você deseja atualizar e o modelo usado para criá-lo. Por exemplo: .

```
Description: Base Template to create Neptune Stack with Engine Version 1.2.0.1 using
 custom Parameter Groups
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBClusterParameterGroup:
    Type: 'AWS::Neptune::DBClusterParameterGroup'
```

```
    Properties:
      Family: neptune1.2
      Description: test-cfn-neptune-db-cluster-parameter-group-description
      Parameters:
        neptune_enable_audit_log: 0
  NeptuneDBParameterGroup:
    Type: 'AWS::Neptune::DBParameterGroup'
    Properties:
      Family: neptune1.2
      Description: test-cfn-neptune-db-parameter-group-description
      Parameters:
        neptune_query_timeout: 20000
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.2.0.1
      DBClusterParameterGroupName:
        Ref: NeptuneDBClusterParameterGroup
    DependsOn:
      - NeptuneDBClusterParameterGroup
  NeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBCluster
      - NeptuneDBParameterGroup
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
      Ref: NeptuneDBCluster
```

Atualize a propriedade `EngineVersion` de `1.2.0.1` para `1.2.0.2`:

```
Description: Template to upgrade minor engine version to 1.2.0.2
Parameters:
  DbInstanceType:
```

```
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBClusterParameterGroup:
    Type: 'AWS::Neptune::DBClusterParameterGroup'
    Properties:
      Family: neptune1.2
      Description: test-cfn-neptune-db-cluster-parameter-group-description
      Parameters:
        neptune_enable_audit_log: 0
  NeptuneDBParameterGroup:
    Type: 'AWS::Neptune::DBParameterGroup'
    Properties:
      Family: neptune1.2
      Description: test-cfn-neptune-db-parameter-group-description
      Parameters:
        neptune_query_timeout: 20000
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.2.0.2
      DBClusterParameterGroupName:
        Ref: NeptuneDBClusterParameterGroup
    DependsOn:
      - NeptuneDBClusterParameterGroup
  NeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBCluster
      - NeptuneDBParameterGroup
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
      Ref: NeptuneDBCluster
```

Agora use AWS CloudFormation para executar o modelo revisado.

Exemplo: atualização da versão principal de 1.1.1.0 para 1.2.0.2 com grupos de parâmetros padrão

Encontre o `DBCluster` que você deseja atualizar e o modelo usado para criá-lo. Por exemplo: .

```
Description: Base Template to create Neptune Stack with Engine Version 1.1.1.0 using
 default Parameter Groups
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.1.1.0
  NeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
    DependsOn:
      - NeptuneDBCluster
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
      Ref: NeptuneDBCluster
```

- Atualize o `DBClusterParameterGroup` padrão para aquele na família de grupos de parâmetros usado pela nova versão do mecanismo (aqui `default.neptune1.2`).
- Para cada `DBInstance` anexado ao `DBCluster`, atualize o `DBParameterGroup` padrão para aquele na família usado pela nova versão do mecanismo (aqui `default.neptune1.2`).
- Defina a propriedade `DBInstanceParameterGroupName` como o grupo de parâmetros padrão nessa família (aqui `default.neptune1.2`).

- Atualize a propriedade `EngineVersion` de `1.1.0.0` para `1.2.0.2`.

O modelo deve ser semelhante ao seguinte:

```
Description: Template to upgrade major engine version to 1.2.0.2 by using upgraded
 default parameter groups
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.2.0.2
      DBClusterParameterGroupName: default.neptune1.2
      DBInstanceParameterGroupName: default.neptune1.2
  NeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName: default.neptune1.2
    DependsOn:
      - NeptuneDBCluster
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
```

Agora use AWS CloudFormation para executar o modelo revisado.

Exemplo: atualização da versão principal de 1.1.1.0 para 1.2.0.2 com grupos de parâmetros personalizados

Encontre o `DBCluster` que você deseja atualizar e o modelo usado para criá-lo. Por exemplo: .

```
Description: Base Template to create Neptune Stack with Engine Version 1.1.1.0 using
 custom Parameter Groups
```

```
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBClusterParameterGroup:
    Type: 'AWS::Neptune::DBClusterParameterGroup'
    Properties:
      Name: engineupgradetestcpg
      Family: neptune1
      Description: 'NeptuneDBClusterParameterGroup with family neptune1'
      Parameters:
        neptune_enable_audit_log: 0
  NeptuneDBParameterGroup:
    Type: 'AWS::Neptune::DBParameterGroup'
    Properties:
      Name: engineupgradetestpg
      Family: neptune1
      Description: 'NeptuneDBParameterGroup1 with family neptune1'
      Parameters:
        neptune_query_timeout: 20000
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.1.1.0
      DBClusterParameterGroupName:
        Ref: NeptuneDBClusterParameterGroup
    DependsOn:
      - NeptuneDBClusterParameterGroup
  NeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBCluster
      - NeptuneDBParameterGroup
Outputs:
  DBClusterId:
```

```
    Description: Neptune Cluster Identifier
    Value:
      Ref: NeptuneDBCluster
```

- Atualize a `DBClusterParameterGroup` família personalizada para aquela usada pela nova versão do mecanismo aqui`default.neptune1.2`).
- Para cada um `DBInstance` anexado ao`DBCluster`, atualize a `DBParameterGroup` família personalizada para aquela usada pela nova versão do mecanismo (aqui`default.neptune1.2`).
- Defina a propriedade `DBInstanceParameterGroupName` como o grupo de parâmetros nessa família (aqui `default.neptune1.2`).
- Atualize a propriedade `EngineVersion` de `1.1.0.0` para `1.2.0.2`.

O modelo deve ser semelhante ao seguinte:

```
Description: Template to upgrade major engine version to 1.2.0.2 by modifying existing
 custom parameter groups
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBClusterParameterGroup:
    Type: 'AWS::Neptune::DBClusterParameterGroup'
    Properties:
      Name: engineupgradetestcpgnew
      Family: neptune1.2
      Description: 'NeptuneDBClusterParameterGroup with family neptune1.2'
      Parameters:
        neptune_enable_audit_log: 0
  NeptuneDBParameterGroup:
    Type: 'AWS::Neptune::DBParameterGroup'
    Properties:
      Name: engineupgradetestpgnew
      Family: neptune1.2
      Description: 'NeptuneDBParameterGroup1 with family neptune1.2'
      Parameters:
        neptune_query_timeout: 20000
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
```

```
    Properties:
      EngineVersion: 1.2.0.2
      DBClusterParameterGroupName:
        Ref: NeptuneDBClusterParameterGroup
      DBInstanceParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBClusterParameterGroup
  NeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBCluster
      - NeptuneDBParameterGroup
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
      Ref: NeptuneDBCluster
```

Agora use AWS CloudFormation para executar o modelo revisado.

Exemplo: atualização da versão principal de 1.1.1.0 para 1.2.0.2 com uma mistura de grupos de parâmetros padrão e personalizados

Encontre o `DBCluster` que você deseja atualizar e o modelo usado para criá-lo. Por exemplo: .

```
Description: Base Template to create Neptune Stack with Engine Version 1.1.1.0 using
 custom Parameter Groups
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBClusterParameterGroup:
    Type: 'AWS::Neptune::DBClusterParameterGroup'
```

```
    Properties:
      Family: neptune1
      Description: 'NeptuneDBClusterParameterGroup with family neptune1'
      Parameters:
        neptune_enable_audit_log: 0
  NeptuneDBParameterGroup:
    Type: 'AWS::Neptune::DBParameterGroup'
    Properties:
      Family: neptune1
      Description: 'NeptuneDBParameterGroup with family neptune1'
      Parameters:
        neptune_query_timeout: 20000
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.1.1.0
      DBClusterParameterGroupName:
        Ref: NeptuneDBClusterParameterGroup
    DependsOn:
      - NeptuneDBClusterParameterGroup
  CustomNeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBCluster
      - NeptuneDBParameterGroup
  DefaultNeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
    DependsOn:
      - NeptuneDBCluster
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
```

```
    Value:
      Ref: NeptuneDBCluster
```

- Para um grupo de parâmetros de cluster personalizados, atualize a família `DBClusterParameterGroup` para aquela correspondente à nova versão do mecanismo, ou seja, `neptune1.2`.
- Para um grupo de parâmetros de cluster padrão, atualize o `DBClusterParameterGroup` para o padrão correspondente à nova versão do mecanismo, ou seja, `default.neptune1.2`.
- Para cada `DBInstance` anexada ao `DBCluster`, atualize um `DBParameterGroup` padrão para o da família usado pela nova versão do mecanismo (aqui `default.neptune1.2`) e um grupo de parâmetros personalizado para um que use a família compatível com a nova versão do mecanismo (aqui `neptune1.2`).
- Defina a propriedade `DBInstanceParameterGroupName` como o grupo de parâmetros na família compatível com a nova versão do mecanismo.

O modelo deve ser semelhante ao seguinte:

```
Description: Template to update Neptune Stack to Engine Version 1.2.0.1 using custom
 and default Parameter Groups
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBClusterParameterGroup:
    Type: 'AWS::Neptune::DBClusterParameterGroup'
    Properties:
      Family: neptune1.2
      Description: 'NeptuneDBClusterParameterGroup with family neptune1.2'
      Parameters:
        neptune_enable_audit_log: 0
  NeptuneDBParameterGroup:
    Type: 'AWS::Neptune::DBParameterGroup'
    Properties:
      Family: neptune1.2
      Description: 'NeptuneDBParameterGroup1 with family neptune1.2'
      Parameters:
        neptune_query_timeout: 20000
  NeptuneDBCluster:
```

```
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.2.0.2
      DBClusterParameterGroupName:
        Ref: NeptuneDBClusterParameterGroup
      DBInstanceParameterGroupName: default.neptune1.2
    DependsOn:
      - NeptuneDBClusterParameterGroup
  CustomNeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBCluster
      - NeptuneDBParameterGroup
  DefaultNeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName: default.neptune1.2
    DependsOn:
      - NeptuneDBCluster
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
      Ref: NeptuneDBCluster
```

Agora use AWS CloudFormation para executar o modelo revisado.

Clonagem de banco de dados no Neptune

Com a clonagem de banco de dados, é possível criar clones de todos os bancos de dados de forma rápida e econômica no Amazon Neptune. Os bancos de dados clonados exigem apenas o espaço adicional mínimo quando criados pela primeira vez. A clonagem de banco de dados usa um protocolo copy-on-write. Os dados são copiados no momento em que são alterados, seja nos bancos de dados de origem ou nos bancos de dados clonados. Você pode fazer vários clones do mesmo cluster de banco de dados. Você também pode criar clones adicionais a partir de outros clones. Para obter mais informações sobre como o protocolo copy-on-write funciona no contexto de armazenamento do Neptune, consulte Protocolo copy-on-write.

Você pode usar a clonagem de banco de dados em uma série de casos de uso, especialmente quando deseja evitar um impacto no seu ambiente de produção, como este:

- Experimente e avalie o impacto das alterações, como alterações de esquema ou no grupo de parâmetros.
- Realize operações com cargas de trabalho intensivas, como exportar dados ou executar consultas analíticas.
- Crie uma cópia de um cluster de banco de dados de produção em um ambiente de não produção para desenvolvimento ou teste.

Para criar um clone de um cluster de banco de dados usando o AWS Management Console

1. Faça login no Console de Gerenciamento da AWS e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. No painel de navegação, escolha Instances (Instâncias). Escolha a instância primária do cluster de banco de dados do qual deseja criar um clone.
3. Selecione Instance actions (Ações de instância) e, em seguida, selecione Create clone (Criar clone).
4. Na página Create Clone (Criar clone), digite um nome para a instância primária do cluster de banco de dados clonado, como o DB instance identifier (Identificador de instância de banco de dados).

   Se desejar, defina quaisquer outras configurações para o cluster de banco de dados clonado. Para obter mais informações sobre as diferentes configurações do cluster de banco de dados, consulte Iniciar usando o console.

5. Escolha Create Clone para iniciar o cluster de banco de dados clonado.

Para criar um clone de um cluster de banco de dados usando o AWS CLI

- Chame o comando restore-db-cluster-to-point-in-time da AWS CLI e forneça os seguintes valores:
  - `--source-db-cluster-identifier`: o nome do cluster de banco de dados de origem do qual será criado um clone.
  - `--db-cluster-identifier`: o nome do cluster de banco de dados clonado.
  - `--restore-type copy-on-write`: o valor `copy-on-write` indica que um cluster de banco de dados clonado deve ser criado.
  - `--use-latest-restorable-time`: especifica que o tempo de backup restaurável mais recente deve ser usado.

> **Note**
>
> O comando restore-db-cluster-to-point-in-time da AWS CLI clona apenas o cluster de banco de dados, e não as instâncias de banco de dados para esse cluster de banco de dados.

O exemplo de Linux/UNIX a seguir cria um clone do cluster do banco de dados `source-db-cluster-id` e nomeia o clone `db-clone-cluster-id`.

```
aws neptune restore-db-cluster-to-point-in-time \
  --region us-east-1 \
  --source-db-cluster-identifier source-db-cluster-id \
  --db-cluster-identifier db-clone-cluster-id \
  --restore-type copy-on-write \
  --use-latest-restorable-time
```

O mesmo exemplo funcionará no Windows se o caractere de escape de fim de linha \ for substituído pelo ^ equivalente do Windows:

```
aws neptune restore-db-cluster-to-point-in-time ^
  --region us-east-1 ^
```

```
  --source-db-cluster-identifier source-db-cluster-id ^
  --db-cluster-identifier db-clone-cluster-id ^
  --restore-type copy-on-write ^
  --use-latest-restorable-time
```

Limitações

A clonagem de banco de dados no Neptune tem as seguintes limitações:

- Não é possível criar clones de bancos de dados em regiões da AWS. Os bancos de dados de clones devem ser criados na mesma região dos bancos de dados de origem.
- Um banco de dados clonado sempre usa o patch mais recente da versão do mecanismo do Neptune que está sendo usada pelo banco de dados do qual foi clonado. Isso ocorre mesmo que o banco de dados de origem ainda não tenha sido atualizado para essa versão de patch. No entanto, a versão do mecanismo em si não muda.
- No momento, o limite é de no máximo 15 clones por cópia do cluster de banco de dados do Neptune, incluindo clones baseados em outros clones. Depois de atingir esse limite, é necessário fazer outra cópia do banco de dados em vez de cloná-lo. No entanto, se você fizer uma nova cópia, ela também poderá ter até 15 clones.
- No momento, a clonagem do banco de dados entre contas não é compatível.
- Você pode fornecer uma nuvem privada virtual (VPC) diferente para seu clone. No entanto, as sub-redes dessas VPCs devem ser mapeadas no mesmo conjunto de Zonas de disponibilidade.

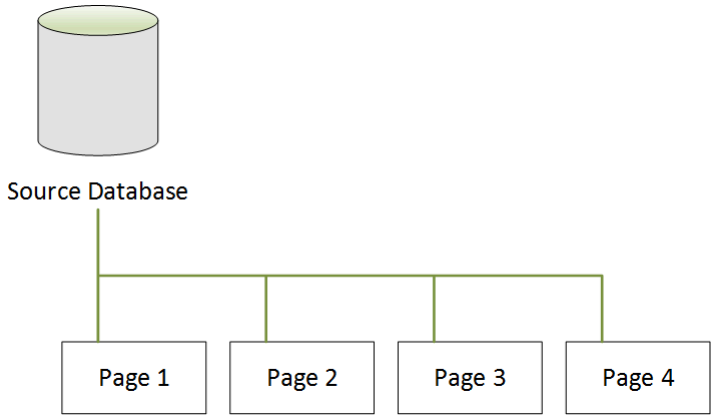
Protocolo copy-on-write para clonagem de banco de dados

Os seguintes cenários ilustram o funcionamento do protocolo copy-on-write.

- Banco de dados do Neptune antes da clonagem
- Banco de dados do Neptune após a clonagem
- Quando uma alteração é feita no banco de dados de origem
- Quando uma alteração é feita no banco de dados clonado

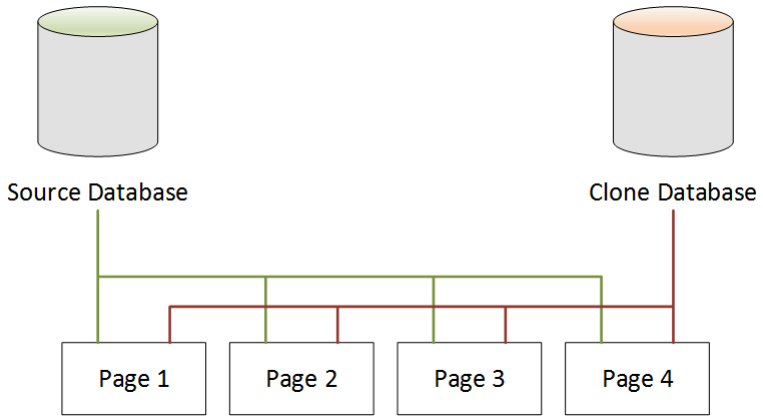
Banco de dados do Neptune antes da clonagem

Os dados em um banco de dados de origem são armazenados em páginas. No diagrama a seguir, o banco de dados de origem apresenta quatro páginas.



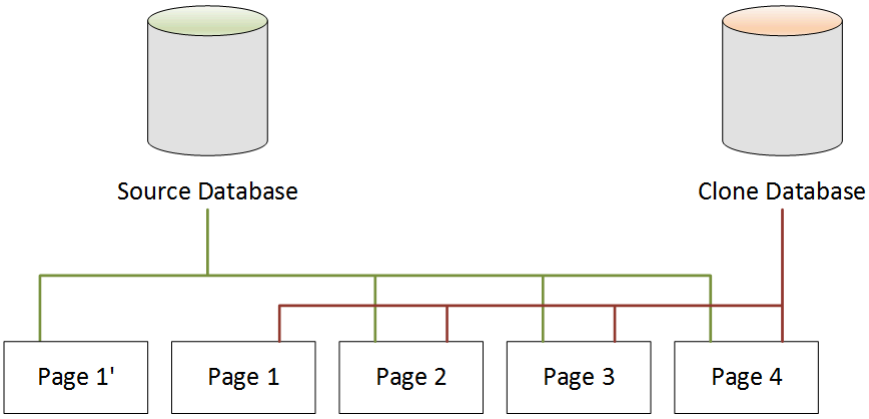
Banco de dados do Neptune após a clonagem

Conforme mostrado no diagrama a seguir, não ocorrem alterações no banco de dados de origem após a clonagem do banco de dados. Tanto o banco de dados de origem quanto o banco de dados clonado apontam para as mesmas quatro páginas. Nenhuma página foi copiada fisicamente, portanto, nenhum armazenamento adicional é necessário.



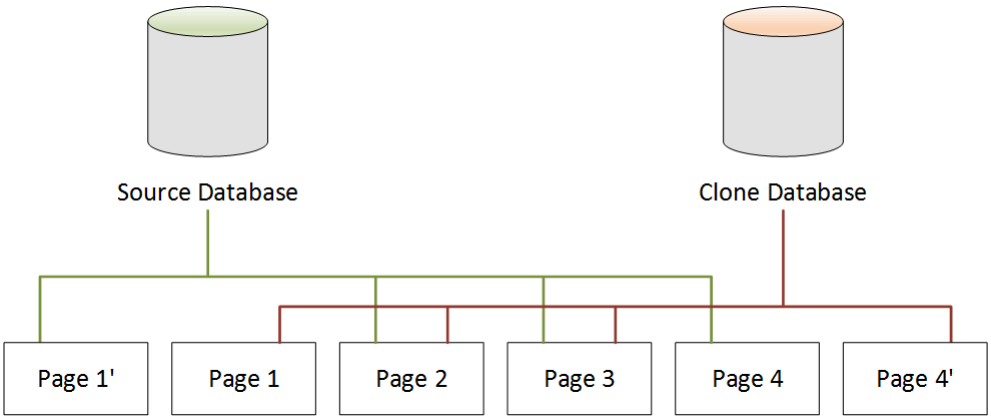
Quando uma alteração é feita no banco de dados de origem

No exemplo a seguir, o banco de dados de origem faz uma alteração nos dados em `Page 1`. Em vez de gravar na `Page 1` original, ele usa um armazenamento adicional para criar uma nova página, chamada `Page 1'`. O banco de dados de origem agora aponta para o novo `Page 1'`, e também para `Page 2`, `Page 3` e `Page 4`. O banco de dados clonado continua a apontar para `Page 1` através da `Page 4`.



Quando uma alteração é feita no banco de dados clonado

No diagrama a seguir, o banco de dados clonado também foi alterado, desta vez em `Page 4`. Em vez de gravar no `Page 4` original, é usado um armazenamento adicional para criar a nova página, chamada `Page 4'`. O banco de dados de origem continua a apontar para `Page 1'`, e também para `Page 2` por meio da `Page 4`, mas o banco de dados clonado agora aponta para `Page 1` por meio da `Page 3` e também para `Page 4'`.



Conforme mostrado no segundo cenário, após a clonagem do banco de dados, não há armazenamento adicional necessário no ponto de criação do clone. No entanto, à medida que ocorrerem mudanças no banco de dados de origem e no banco de dados clonado, somente as páginas alteradas serão criadas, conforme mostrado nos terceiro e quarto cenários. À medida que ocorrerem mais alterações ao longo do tempo no banco de dados de origem e no banco de dados clonado, você precisará de mais armazenamento incremental para captar e armazenar as alterações.

Excluir um banco de dados de origem

A exclusão de um banco de dados de origem não afeta os bancos de dados clonados a ele associados. Os bancos de dados clonados continuam a apontar para as páginas que pertenciam anteriormente ao banco de dados de origem.

Gerenciar instâncias do Amazon Neptune

As seções a seguir têm informações sobre operações no nível da instância.

Tópicos

Classe de instância de intermitência T3 do Neptune.

Além das classes de instância de desempenho fixo, como R5 e R6, o Amazon Neptune oferece a opção de usar uma instância T3 de desempenho de intermitência. Ao desenvolver a aplicação de grafos, convém que o banco de dados seja rápido e responsivo, mas não precisa usá-lo o tempo todo. A classe de instância db.t3.medium do Neptune é exatamente o que você deve usar nessa situação, por um custo significativamente menor do que a classe de instância de desempenho fixo mais barata.

Uma instância de intermitência é executada no nível da linha de base do desempenho da CPU até que uma carga de trabalho precise de mais e gera picos acima da linha de base pelo tempo necessário para a carga de trabalho. O preço por hora cobre os picos, desde que o uso médio da CPU não ultrapasse a linha de base por um período de 24 horas. Para a maioria das situações de desenvolvimento e teste, isso significa bom desempenho a um baixo custo.

Se você começar com uma classe de instância T3, poderá alternar facilmente para uma instância de desempenho fixo posteriormente quando estiver tudo pronto para entrar em produção, usando o AWS Management Console, a AWS CLI ou um dos AWS SDKs.

A intermitência T3 é governada por créditos da CPU

Um crédito da CPU representa o uso total de um núcleo de CPU virtual (vCPU) por um minuto. Isso também significa 50% de uso de uma vCPU por dois minutos, ou 25% de uso de duas vCPUs por dois minutos, e assim por diante.

Uma instância T3 acumula créditos da CPU quando está ociosa e os usa quando está ativa, ambos medidos em resolução de milissegundos. A classe de instância db.t3.medium tem duas vCPUs, e cada uma recebe 12 créditos da CPU por hora quando ociosa. Isso significa que 20% do uso de cada vCPU resulta em um saldo de crédito da CPU igual a zero. Os 12 créditos da CPU recebidos são gastos em 20% do uso da vCPU (pois 20% de 60 minutos também é 12). Desta forma, esse uso de 20% é a taxa de utilização da linha de base que gera um saldo de crédito da CPU nem positivo nem negativo.

O tempo ocioso (utilização da CPU abaixo de 20% do total disponível) faz com que os créditos da CPU sejam armazenados em um bucket de saldo de crédito, até o limite para uma classe de instância db.t3.medium de 576 (o número máximo de créditos da CPU que podem ser acumulados em 24 horas, ou seja 2 x 12 x 24). Acima desse limite, os créditos da CPU são simplesmente descartados.

Quando necessário, a utilização da CPU pode ter picos de até 100% pelo tempo necessário para a carga de trabalho, mesmo após o saldo de crédito da CPU ficar abaixo de zero. Se a instância manter um saldo negativo continuamente por 24 horas, ela incorrerá uma cobrança adicional de 0,05 USD para cada -60 créditos da CPU acumulados nesse período. No entanto, para a maioria das cargas de trabalho de desenvolvimento e teste, a intermitência é geralmente coberta pelo tempo ocioso antes ou após o pico.

> **Note**
>
> A classe de instância T3 do Neptune é configurada como o modo ilimitado do Amazon EC2.

Usar o AWS Management Console para criar uma instância de intermitência T3

No AWS Management Console, é possível criar uma instância primária de cluster de banco de dados, ou uma instância de réplica de leitura que usa a classe da instância `db.t3.medium`, ou você pode modificar uma instância existente para usar a classe de instância `db.t3.medium`.

Por exemplo, para criar uma instância principal do cluster de banco de dados no console do Neptune:

- Escolha Create Database (Criar banco de dados).
- Escolha uma DB engine version (Versão do mecanismo de banco de dados) igual ou posterior a `1.0.2.2`.
- Em Purpose (Finalidade), escolha Development and Testing (Desenvolvimento e teste).
- Para a DB instance class (Classe da instância de banco de dados), aceite o padrão: `db.t3.medium — 2 vCPU, 4 GiB RAM`.

Usar o AWS CLI para criar uma instância de intermitência T3

Também é possível usar a AWS CLI para fazer a mesma coisa:

```
aws neptune create-db-cluster \
    --db-cluster-identifier (name for a new DB cluster) \
    --engine neptune \
    --engine-version "1.0.2.2"

aws neptune create-db-instance \
```

```
    --db-cluster-identifier (name of the new DB cluster) \
    --db-instance-identifier (name for the primary writer instance in the cluster) \
    --engine neptune \
    --db-instance-class db.t3.medium
```

Modificar uma instância de banco de dados do Neptune (e aplicar imediatamente)

É possível aplicar a maioria das alterações em uma instância de banco de dados do Amazon Neptune imediatamente ou adiá-las até a próxima janela de manutenção. Algumas modificações, como alterações em grupos de parâmetros, exigem que você reinicie manualmente sua instância de banco de dados para que a alteração entre em vigor.

> **Important**
>
> Modificações geram uma interrupção, pois o Neptune deve reiniciar a instância de banco de dados para que a alteração seja aplicada. Analise o impacto sobre o banco de dados e as aplicações antes de modificar as configurações da instância de banco de dados.

Configurações comuns e implicações do tempo de inatividade

A tabela a seguir contém detalhes sobre quais configurações você pode modificar, quando as alterações podem ser aplicadas e se as alterações causam tempo de inatividade para a instância de banco de dados.

Configuração da instância de banco de dados	Observações sobre tempo de inatividade	
Classe de instância de banco de dados	Ocorre uma interrupção durante essa alteração, seja ela aplicada imediatamente ou durante a próxima janela de manutenção.	
DB instance identifier	A instância de banco de dados é reinicializada e ocorre uma interrupção durante essa alteração, seja ela aplicada imediatamente ou durante a próxima janela de manutençã o.	

Configuração da instância de banco de dados	Observações sobre tempo de inatividade	
Subnet group	A instância de banco de dados é reinicializada e ocorre uma interrupção durante essa alteração, seja ela aplicada imediatamente ou durante a próxima janela de manutenção.	
Grupo de segurança	A alteração é aplicada de forma assíncrona o mais rápido possível, independentemente de quando você especifica que as alterações devem ocorrer, sem resultados de interrupção.	–
Certificate Authority (Autoridade certificadora)	Por padrão, a instância de banco de dados é reiniciada quando você atribui uma nova autoridade certificadora.	
Database Port	A alteração sempre ocorre imediatamente, fazendo com que a instância de banco de dados seja reinicializada e ocorra uma interrupção.	

Configuração da instância de banco de dados	Observações sobre tempo de inatividade	
Grupo de parâmetros de banco de dados	Alterar essa configuração não resultará em uma interrupção. O nome do parameter group propriamente dito é alterado imediatamente, mas as alterações de parâmetros reais não serão aplicadas até que você reinicialize a instância sem failover. Nesse caso, a instância de banco de dados não será reinicializada automaticamente, e as alterações de parâmetro não serão aplicadas durante a próxima janela de manutenção. No entanto, se você modificar parâmetros dinâmicos no grupo de parâmetros de banco de dados recém-associado, essas alterações serão aplicadas imediatamente sem uma reinicialização. Para obter mais informações, consulte Reinicializar uma instância de banco de dados no Amazon Neptune.	
Grupo de parâmetros do cluster de banco de dados	O grupo de parâmetros de banco de dados é alterado imediatamente.	

Configuração da instância de banco de dados	Observações sobre tempo de inatividade	
Backup retention period (Período de retenção de backup)	Se você especificar que as alterações devem ocorrer imediatamente, essa alteração ocorrerá imediatamente. Caso contrário, se você alterar a configuração de um valor diferente de zero para outro valor diferente de zero, a alteração será aplicada de forma assíncrona o mais rápido possível. Qualquer outra alteração ocorrerá durante a próxima janela de manutenção. Uma falha ocorrerá se você alterar de zero para um valor diferente de zero ou de um valor diferente de zero para zero.	
Log de auditoria	Selecione Log de auditoria se quiser usar o registro em log de auditoria por meio do CloudWatch Logs. Você também deve definir o parâmetro `neptune_enable_audit_log` no grupo de parâmetros de cluster de banco de dados como `enable` (1) para que o registro em log de auditoria seja habilitado.	

Configuração da instância de banco de dados	Observações sobre tempo de inatividade	
Atualização da versão secundária automática	Selecione Habilitar a atualizaç ão automática da versão secundária se quiser permitir que o cluster de banco de dados do Neptune receba atualizações de versão secundária do mecanismo quando disponíveis. A opção Atualização automática de versão secundária só se aplica a atualizações para versões secundárias do mecanismo do cluster de banco de dados do Amazon Neptune. Ela não se aplica a patches regulares aplicados para manter a estabilidade do sistema.	

Renomear uma instância de banco de dados do Neptune

É possível renomear uma instância de banco de dados do Amazon Neptune usando o AWS Management Console. Renomear uma instância de banco de dados pode ter efeitos de longo alcance. Veja a seguir uma lista de coisas que você deve saber antes de renomear uma instância de banco de dados.

- Quando você renomeia uma instância de banco de dados, o endpoint dessa instância é alterado, pois a URL inclui o nome atribuído a ela. Você sempre deve redirecionar o tráfego da URL antiga para a nova.
- Ao renomear uma instância de banco de dados, o nome DNS antigo que era usado pelo cluster de banco de dados é excluído imediatamente, mas pode permanecer armazenado em cache por alguns minutos. O novo nome DNS da instância de banco de dados renomeada torna-se efetivo em cerca de 10 minutos. A instância de banco de dados renomeada não ficará disponível até que o novo nome entre em vigor.
- Você não pode usar um nome de instância de banco de dados existente ao renomear uma instância.
- Todas as réplicas de leitura associadas a uma instância de banco de dados permanecem associadas a essa instância depois que ela é renomeada. Por exemplo, suponha que você tem uma instância de banco de dados que atende seu banco de dados de produção, e que essa instância tem várias réplicas de leitura associadas. Se você renomear a instância de banco de dados e a substituir no ambiente de produção por um DB snapshot, a instância de banco de dados que você renomeou ainda terá as réplicas de leitura associadas a ela.
- As métricas e os eventos associados ao nome de uma instância de banco de dados serão mantidos se você reutilizar um nome de instância de banco de dados. Por exemplo, se você promover uma réplica de leitura e a renomear para que ela tenha o nome da instância principal anterior, os eventos e as métricas associados à instância principal serão associados à instância renomeada.
- Tags de instâncias de bancos de dados permanecem com a instância de banco de dados, independentemente de renomeação.
- snapshot de banco de dadoss são mantidos para uma instância de banco de dados renomeada.

Como renomear uma instância de banco de dados usando o console do Neptune

1. Faça login no Console de Gerenciamento da AWS e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.

2. No painel de navegação, escolha Databases (Bancos de dados).
3. Selecione o botão de opção ao lado da instância de banco de dados que deseja renomear.
4. No menu Instance actions (Ações da instância), escolha Modify (Modificar).
5. Insira um novo nome na caixa de texto DB Instance Identifier (Identificador da instância de banco de dados). Selecione Apply immediately (Aplicar imediatamente) e, em seguida, selecione Continue (Continuar).
6. Selecione Modify DB Instance (Modificar instância de banco de dados) para concluir a alteração.

Reinicializar uma instância de banco de dados no Amazon Neptune

Em alguns casos, se você modificar uma instância de banco de dados do Amazon Neptune, alterar o grupo de parâmetros de banco de dados associado à instância ou alterar um parâmetro de banco de dados estático em um grupo de parâmetros usado pela instância, será necessário reinicializar a instância para que as alterações sejam aplicadas.

Reiniciar uma instância de banco de dados reinicia o serviço de mecanismo de banco de dados. Uma reinicialização também aplica à instância de banco de dados todas as alterações do parameter group de banco de dados associado que estavam pendentes. Reinicializar uma instância de banco de dados resulta em uma interrupção momentânea da instância, durante a qual o status da instância de banco de dados é definido como rebooting. Se a instância do Neptune estiver configurada para multi-AZ, a reinicialização poderá ser realizada por meio de failover. Um evento do Neptune é criado quando a reinicialização é concluída.

Se a instância de banco de dados for uma implantação Multi-AZ, você poderá forçar um failover de uma zona de disponibilidade para outra ao selecionar a opção Reboot (Reinicializar). Ao forçar um failover da instância de banco de dados, o Neptune automaticamente alterna para uma réplica em espera em outra zona de disponibilidade. Em seguida, atualiza o registro DNS da instância de banco de dados para apontar para a instância de banco de dados em espera. Como resultado, você deve limpar e restabelecer todas as conexões existentes com a instância de banco de dados.

A opção Reboot with failover (Reinicialização com failover) é benéfica quando você deseja simular uma falha de uma instância de banco de dados para testes ou restaurar as operações na zona de disponibilidade original após a ocorrência de um failover. Para obter mais informações, consulte High Availability (Multi-AZ) no Guia do usuário do Amazon RDS. Quando você reinicializa um cluster de banco de dados, ele executa failover na réplica em espera. A reinicialização de uma réplica do Neptune não inicia um failover.

O tempo necessário para reinicializar depende do processo de recuperação de falhas. Para melhorar o tempo de reinicialização, recomendamos reduzir as atividades do banco de dados o máximo possível durante o processo de reinicialização para reduzir a atividade de reversão para transações em trânsito.

No console, a opção Reboot (Reiniciar) poderá estar desabilitada se a instância de banco de dados não estiver no estado Available (Disponível). Isso pode ser devido a vários motivos, como um backup em andamento, uma modificação solicitada pelo cliente ou uma ação da janela de manutenção.

> **Note**
>
> Antes de Versão: 1.2.0.0 (21/07/2022), todas as réplicas de leitura em um cluster de banco de dados eram reinicializadas automaticamente quando a instância (de gravador) principal era reiniciada.
> A partir de Versão: 1.2.0.0 (21/07/2022), reiniciar a instância principal não faz com que as réplicas sejam reiniciadas. Isso significa que, se você estiver alterando um parâmetro de cluster, deverá reiniciar cada instância separadamente para receber a alteração do parâmetro (consulte Grupos de parâmetros).

Como reinicializar uma instância de banco de dados usando o console do Neptune

1. Faça login no Console de Gerenciamento da AWS e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. No painel de navegação, escolha Databases (Bancos de dados).
3. Selecione a instância de banco de dados que você deseja reinicializar.
4. Escolha Instance actions e depois Reboot.
5. Para forçar um failover de uma zona de disponibilidade para outra, selecione Reboot with failover? (Reinicializar com failover?) na caixa de diálogo Reboot DB Instance (Reinicializar instância de banco de dados).
6. Escolha Reboot. Para cancelar a reinicialização, selecione Cancel (Cancelar).

Excluir uma instância de banco de dados no Amazon Neptune

É possível excluir uma instância de banco de dados do Amazon Neptune em qualquer estado e a qualquer momento, desde que a instância tenha sido iniciada e a proteção contra exclusão esteja desabilitada na instância.

Não será possível excluir uma instância de banco de dados se a proteção contra exclusão estiver habilitada

É possível excluir somente instâncias que tenham a proteção contra exclusão desabilitada. O Neptune impõe a proteção contra exclusão independentemente de usar o console, a AWS CLI ou as APIs para excluir uma instância de banco de dados.

Por padrão, a proteção contra exclusão permanece habilitada quando é criada uma instância de banco de dados de produção usando o AWS Management Console.

A proteção contra exclusão será desabilitada por padrão se você usar os comandos da AWS CLI ou da API para criar uma instância de banco de dados.

Para excluir uma instância de banco de dados que tenha proteção de exclusão habilitada, primeiro modifique a instância para definir seu campo `DeletionProtection` como `false`.

A ativação ou a desativação da proteção contra exclusão não causa uma interrupção.

Criar um snapshot final de sua instância de banco de dados antes de excluí-la

Para excluir uma instância de banco de dados, você deve especificar o nome da instância e se deseja obter um DB snapshot final da instância. Se a instância de banco de dados que está sendo excluída tiver um status de Creating (Criando), você não poderá obter um DB snapshot final. Se a instância de banco de dados estiver em estado de falha com um status failed (com falha), incompatible-restore (restauração incompatível) ou incompatible-network (rede incompatível), você só poderá excluir a instância quando o parâmetro `SkipFinalSnapshot` estiver definido como `true`.

Se você excluir todas as instâncias de banco de dados do Neptune em um cluster de banco de dados usando o AWS Management Console, o cluster de banco de dados será excluído automaticamente. Se estiver usando o SDK ou a AWS CLI, você deverá excluir o cluster de banco de dados manualmente depois de excluir a última instância.

> **Important**
>
> Se você excluir todo um cluster de banco de dados, todos os seus backups automatizados serão excluídos ao mesmo tempo e não poderão ser recuperados. Isso significa que, a menos que você opte por criar um snapshot de banco de dados final manualmente, não poderá restaurar a instância de banco de dados para seu estado final posteriormente. Os snapshots manuais de uma instância não são excluídos quando o cluster é excluído.

Se a instância de banco de dados que deseja excluir tiver uma réplica de leitura, você deverá promover a réplica de leitura ou excluí-la.

Nos exemplos a seguir, você exclui uma instância de banco de dados com e sem um DB snapshot final.

Excluir uma instância de banco de dados sem um snapshot final

Se desejar excluir a instância de banco de dados rapidamente, você poderá ignorar a criação de um DB snapshot final. Quando você exclui uma instância de banco de dados, todos os backups automatizados são excluídos e não podem ser recuperados. Os snapshots manuais não são excluídos.

Como excluir uma instância de banco de dados sem um snapshot de banco de dados final usando o console do Neptune

1. Faça login no Console de Gerenciamento da AWS e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. No painel de navegação, escolha Databases (Bancos de dados).
3. Na lista Instances (Instâncias), selecione o botão de opção ao lado da instância de banco de dados que você deseja excluir.
4. Escolha Instance actions e, em seguida, escolha Delete.
5. Escolha No (Não) na caixa Create final snapshot? (Criar snapshot final?).
6. Escolha Delete (Excluir).

Excluir uma instância de banco de dados com um snapshot final

Se desejar poder restaurar a instância de banco de dados excluída posteriormente, você poderá criar um DB snapshot final. Todos os backups automáticos também serão excluídos e não poderão ser recuperados. Os snapshots manuais não são excluídos.

Como excluir uma instância de banco de dados com um snapshot de banco de dados final usando o console do Neptune

1. Faça login no Console de Gerenciamento da AWS e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. No painel de navegação, escolha Databases (Bancos de dados).
3. Na lista Instances (Instâncias), selecione o botão de opção ao lado da instância de banco de dados que você deseja excluir.
4. Escolha Instance actions e, em seguida, escolha Delete.
5. Escolha Yes (Sim) na caixa Create final snapshot? (Criar snapshot final?).
6. Na caixa Final Snapshot name (Nome do snapshot final), digite o nome de seu DB snapshot final.
7. Escolha Delete (Excluir).

Você pode verificar a integridade de uma instância, determinar o seu tipo, descobrir qual versão de mecanismo você tem instalada no momento e obter outras informações sobre uma instância usando a API de status da instância.

Amazon Neptune Serverless

O Amazon Neptune Serverless é uma configuração de ajuste de escala automático sob demanda projetada para escalar o cluster de banco de dados conforme necessário a fim de atender até mesmo os aumentos muito grandes na demanda de processamento e depois reduzir a escala verticalmente quando a demanda diminui. Isso ajuda a automatizar os processos de monitoramento da workload e o ajuste da capacidade do banco de dados do Neptune. A capacidade é ajustada automaticamente com base na demanda da aplicação e você é cobrado apenas pelos recursos de que a aplicação precisa.

Casos de uso do Neptune Serverless

O Neptune Serverless é compatível com muitos tipos de workloads. É adequado para workloads exigentes e altamente variáveis e poderá ser muito útil se o uso do banco de dados for geralmente intenso por curtos intervalos de tempo, seguidos por longos períodos de atividade leve ou nenhuma atividade. O Neptune Serverless é útil principalmente para os seguintes casos de uso:

- Workloads variáveis: workloads que têm aumentos repentinos e imprevisíveis na atividade da CPU. Com o Neptune Serverless, o banco de dados de grafos escala a capacidade automaticamente para atender às necessidades da workload e torna a reduz a escala da capacidade verticalmente quando o pico da atividade termina. Não é mais necessário provisionar para capacidade de pico ou média. É possível especificar um limite de capacidade superior para lidar com os picos das workloads, e essa capacidade não é usada, a menos que seja necessária.

  A granularidade da escalabilidade fornecida pelo Neptune Serverless ajuda a combinar a capacidade às necessidades da workload. O Neptune Serverless pode adicionar ou remover capacidade em incrementos refinados com base no que é necessário. Ele pode adicionar apenas metade de uma Unidade de capacidade do Neptune (NCU) quando apenas um pouco mais de capacidade é necessária.
- Aplicações multilocatárias: ao aproveitar as vantagens do Neptune Serverless, é possível criar um cluster de banco de dados separado para cada uma das aplicações que você precisa executar sem precisar gerenciar esses clusters de locatários individualmente. Cada um dos clusters de locatários pode ter diferentes períodos de atividade intensa e de inatividade, dependendo de vários fatores, mas o Neptune Serverless pode escalá-los de forma eficiente sem sua intervenção.
- Novas aplicações: quando você implanta uma nova aplicação, geralmente não tem certeza de quanta capacidade de banco de dados será necessária. Usando o Neptune Serverless, é possível

configurar um cluster de banco de dados que pode ser escalado automaticamente para atender aos requisitos de capacidade da nova aplicação à medida que ela se desenvolve.

- Planejamento de capacidade: suponha que você geralmente ajuste a capacidade do banco de dados ou verifique a capacidade ideal do banco de dados para a workload, modificando as classes de todas as instâncias de banco de dados em um cluster. Com o Neptune Serverless, é possível evitar essa sobrecarga administrativa. Em vez disso, você pode modificar as instâncias de banco de dados existentes de provisionadas para sem servidor ou de sem servidor para provisionadas sem precisar criar uma instância ou um cluster de banco de dados.
- Desenvolvimento e teste: o Neptune Serverless também é perfeito para ambientes de desenvolvimento e teste. Com o Neptune Serverless, é possível criar instâncias de banco de dados com uma capacidade máxima alta o suficiente para testar a aplicação mais exigente e uma capacidade mínima baixa para todos os outros momentos em que o sistema pode ficar ocioso entre os testes.

O Neptune Serverless só escala a capacidade computacional. O volume de armazenamento permanece o mesmo e não é afetado pela escalabilidade sem servidor.

> **Note**
>
> Você também pode usar o ajuste de escala automático do Neptune com o Neptune Serverless para lidar com diferentes tipos de variações da workload.

Restrições do Amazon Neptune Serverless

- Não disponível em todas as regiões: o Neptune Serverless só está disponível nas seguintes regiões:
  - Leste dos EUA (Norte da Virgínia): `us-east-1`
  - Leste dos EUA (Ohio): `us-east-2`
  - Oeste dos EUA (N. da Califórnia): `us-west-1`
  - Oeste dos EUA (Oregon): `us-west-2`
  - Canadá (Central): `ca-central-1`
  - Europa (Estocolmo): `eu-north-1`
  - Europa (Irlanda): `eu-west-1`
  - Europa (Londres): `eu-west-2`

  - Europa (Frankfurt): `eu-central-1`
  - Ásia-Pacífico (Tóquio): `ap-northeast-1`
  - Ásia-Pacífico (Singapura): `ap-southeast-1`
  - Ásia-Pacífico (Sydney): `ap-southeast-2`
- Não disponível nas versões anteriores do mecanismo: o Neptune Serverless só está disponível nas versões 1.2.0.1 ou posterior do mecanismo.
- Não compatível com o cache de pesquisa do Neptune: o cache de pesquisa não funciona com instâncias de banco de dados sem servidor.
- A memória máxima em uma instância sem servidor é 256 GB: a configuração de `MaxCapacity` de 128 NCUs (a configuração mais alta compatível) permite que uma instância do Neptune Serverless seja escalada para 256 GB de memória, o que é equivalente a de um tipo de instância provisionada `R6g.8XL`.

Escalabilidade da capacidade em um cluster de banco de dados do Neptune Serverless

A configuração de um cluster de banco de dados do Neptune Serverless é semelhante à configuração de um cluster provisionado normal, com configuração adicional de unidades mínimas e máximas de escalabilidade e com o tipo de instância definido como `db.serverless`. A configuração de escalabilidade é definida em unidades de capacidade do Neptune (NCUs), e cada uma consiste em 2 GiB (gibibyte) de memória (RAM), além da capacidade do processador virtual (vCPU) e da rede associadas. É definida como parte de um objeto `ServerlessV2ScalingConfiguration`, representado em JSON da seguinte forma:

```
"ServerlessV2ScalingConfiguration": {
  "MinCapacity": (minimum NCUs, a floating-point number such as   1.0),
  "MaxCapacity": (maximum NCUs, a floating-point number such as 128.0)
}
```

A qualquer momento, cada instância de gravador ou de leitor do Neptune tem uma capacidade medida por um número de ponto flutuante que representa o número de NCUs que estão sendo usadas no momento por essa instância. Você pode usar a métrica CloudWatch ServerlessDatabaseCapacidade em um nível de instância para descobrir quantas NCUs uma determinada instância de banco de dados está usando atualmente e a métrica NCUUtilization para descobrir qual porcentagem de sua capacidade máxima a instância está usando. As duas métricas

também estão disponíveis em um nível de cluster de banco de dados para mostrar a utilização média de recursos para o cluster de banco de dados como um todo.

Ao criar um cluster de banco de dados do Neptune Serverless, você define o número mínimo e máximo de unidades de capacidade do Neptune (NCUs) para todas as instâncias sem servidor.

O valor mínimo de NCU especificado define o menor tamanho para o qual uma instância sem servidor no cluster de banco de dados pode ser reduzida e, da mesma forma, o valor máximo de NCU estabelece o maior tamanho até o qual uma instância sem servidor pode se expandir. O valor máximo mais alto de NCU que você pode definir é 128,0 NCUs e o mínimo mais baixo é 1,0 NCUs.

O Neptune monitora continuamente a carga em cada instância do Neptune Serverless monitorando a utilização de recursos, como CPU, memória e rede. A carga é gerada pelas operações do banco de dados da aplicação, pelo processamento em segundo plano do servidor e por outras tarefas administrativas.

Quando a carga em uma instância sem servidor atinge o limite da capacidade atual ou quando o Neptune detecta qualquer outro problema de desempenho, a escala da instância é aumentada verticalmente de modo automático. Quando a carga na instância diminui, a escala da capacidade é reduzida verticalmente para as unidades de capacidade mínima configuradas, com a capacidade da CPU sendo liberada antes da memória. Essa arquitetura permite a liberação de recursos de forma controlada e reduz as flutuações de demanda com eficácia.

É possível escalar uma instância de leitor junto com a instância de escritor ou escalar de forma independente definindo o nível de promoção. As instâncias de leitor nos níveis de promoção 0 e 1 são escaladas ao mesmo tempo que o gravador, o que as mantém dimensionadas na capacidade certa para assumir a workload do gravador de forma rápida em caso de failover. Os leitores nos níveis de promoção de 2 a 15 são escalados independentemente da instância de gravador e uns dos outros.

Se você criou o cluster de banco de dados do Neptune como um cluster multi-AZ para garantir alta disponibilidade, o Neptune Serverless aumenta e reduz a escala das instâncias verticalmente em todas as AZs com a carga do banco de dados. É possível definir o nível de promoção de uma instância de leitor em uma AZ secundária como 0 ou 1 para que a escala seja aumentada ou reduzida verticalmente junto com a capacidade da instância de gravador na AZ principal, para que ela esteja pronta para assumir a workload atual a qualquer momento.

> **Note**
>
> O armazenamento de um cluster de banco de dados do Neptune consiste em seis cópias de todos os dados, distribuídos por três AZs, independentemente de você ter ou não criado o cluster como um cluster multi-AZ. A replicação do armazenamento é tratada pelo subsistema de armazenamento e não é afetada pelo Neptune Serverless.

Selecionar um valor mínimo de capacidade para um cluster de banco de dados do Neptune Serverless

O menor valor que você pode definir para a capacidade mínima é `1.0` NCUs.

Não defina o valor mínimo abaixo do que a aplicação exige para funcionar com eficiência. Configurá-lo como um valor muito baixo pode gerar uma taxa maior de tempos limite em determinadas workloads que consomem muita memória.

Definir o valor mínimo o mais baixo possível pode gerar redução de custos, pois o cluster usará recursos mínimos quando a demanda for baixa. No entanto, se a workload tende a flutuar drasticamente, de muito baixa para muito alta, convém definir o mínimo como um valor mais alto, porque um mínimo mais alto permite que a escala das instâncias do Neptune Serverless seja aumentada verticalmente com maior rapidez.

O motivo disso é que o Neptune escolhe incrementos de escala com base na capacidade atual. Se a capacidade atual for baixa, o Neptune inicialmente aumentará a escala verticalmente de forma lenta. Se o mínimo for maior, o Neptune começará com um incremento de escala maior e, portanto, a escala poderá ser aumentada verticalmente com maior rapidez para atender a um grande aumento repentino na workload.

Selecionar um valor máximo de capacidade para um cluster de banco de dados do Neptune Serverless

O maior valor que você pode definir para a capacidade máxima é `128.0` NCUs, e o menor valor que é possível definir para a capacidade máxima é `2.5` NCUs. Qualquer valor máximo de capacidade definido deve ser pelo menos tão grande quanto o valor mínimo de capacidade definido.

Como regra, defina o valor máximo alto o suficiente para lidar com o pico de carga que a aplicação provavelmente encontrará. Configurá-lo como um valor muito baixo pode gerar uma taxa maior de tempos limite em determinadas workloads que consomem muita memória.

Definir o valor máximo o mais alto possível tem a vantagem de que a aplicação provavelmente poderá lidar até mesmo com as workloads mais inesperadas. A desvantagem é que você perde a capacidade de prever e controlar os custos dos recursos. Um aumento inesperado na demanda pode acabar custando muito mais do que o previsto pelo orçamento.

A vantagem de um valor máximo cuidadosamente definido é que ele permite atender aos picos de demanda e, ao mesmo tempo, limitar os custos de computação do Neptune.

> **Note**
>
> A alteração do intervalo de capacidade de um cluster de banco de dados do Neptune Serverless causa alterações nos valores padrão de alguns parâmetros de configuração. O Neptune pode aplicar alguns desses novos padrões imediatamente, mas algumas das alterações de parâmetros dinâmicos são aplicadas somente após a reinicialização. O status de `pending-reboot` indica que é necessária uma reinicialização para aplicar algumas alterações de parâmetro.

Use a configuração existente para estimar os requisitos da tecnologia sem servidor

Se você normalmente modifica a classe das instâncias de banco de dados provisionadas para atender a uma workload excepcionalmente alta ou baixa, é possível usar essa experiência para realizar uma estimativa aproximada do intervalo de capacidade do Neptune equivalente.

Estimar a melhor configuração de capacidade mínima

É possível aplicar as informações sobre o cluster de banco de dados do Neptune existente para estimar a configuração de capacidade mínima sem servidor que funcionará melhor.

Por exemplo, se a workload provisionada tiver requisitos de memória muito altos para classes de instâncias de banco de dados pequenas, como `T3` ou `T4g`, selecione uma configuração mínima de NCU que forneça memória comparável a uma instância de banco de dados `R5` ou `R6g`.

Ou suponha que você use a classe de instância de banco de dados `db.r6g.xlarge` quando o cluster tem uma workload baixa. A classe de instância de banco de dados tem 32 GiB de memória, então é possível especificar uma configuração mínima de NCU de 16 para criar instâncias sem servidor cuja escala possa ser reduzida verticalmente para aproximadamente a mesma capacidade

(cada NCU corresponde a cerca de 2 GiB de memória). Se a instância `db.r6g.xlarge` às vezes for subutilizada, talvez seja possível especificar um valor menor.

Se a aplicação funcionar de forma mais eficiente quando as instâncias de banco de dados puderem manter uma quantidade de dados específica na memória ou no cache de buffer, pense em especificar uma configuração mínima de NCU grande o suficiente para fornecer memória adequada para isso. Caso contrário, os dados poderão ser removidos do cache do buffer quando a escala das instâncias sem servidor for reduzida verticalmente e precisarão ser lidos novamente no cache do buffer ao longo do tempo, quando a escala das instâncias voltar a aumentar verticalmente. Se a quantidade de E/S para trazer os dados de volta para o cache de buffer for substancial, talvez seja válido escolher um valor mínimo de NCU mais alto.

Se você descobrir que as instâncias sem servidor estão sendo executadas a maior parte do tempo em uma capacidade específica, é bom definir a capacidade mínima um pouco abaixo disso. O Neptune Serverless pode estimar com maior eficiência quanto e com que rapidez a escala será aumentada verticalmente quando a capacidade atual não for drasticamente menor do que a capacidade necessária.

Em uma configuração mista, com um gravador provisionado e leitores do Neptune Serverless, os leitores não poderão ser escalados junto com o gravador. Como eles são escalados de forma independente, definir uma capacidade mínima baixa para eles pode ocasionar atraso excessivo de replicação. Eles podem não ter capacidade suficiente para acompanhar as alterações que o gravador estiver fazendo quando houver uma workload com uso altamente intenso de gravação. Nessa situação, defina uma capacidade mínima que seja comparável à capacidade do gravador. Especificamente, se você observar atraso da réplica nos leitores que estão nos níveis de promoção 2 a 15, aumente a configuração de capacidade mínima do cluster.

Estimar a melhor configuração de capacidade máxima

Também é possível aplicar as informações sobre o cluster de banco de dados do Neptune existente para estimar a configuração de capacidade máxima sem servidor que funcionará melhor.

Por exemplo, suponha que você use a classe de instância de banco de dados `db.r6g.4xlarge` quando o cluster tem uma workload alta. Essa classe de instância de banco de dados tem 128 GiB de memória, então você pode especificar uma configuração máxima de NCU de 64 para configurar instâncias equivalentes do Neptune Serverless (cada NCU corresponde a cerca de 2 GiB de memória). É possível especificar um valor maior para permitir que a escala da instância de banco de dados seja aumentada verticalmente ainda mais no caso de a instância `db.r6g.4xlarge` nem sempre conseguir lidar com a workload.

Se picos inesperados na workload forem raros, pode fazer sentido definir a capacidade máxima alta o suficiente para manter o desempenho da aplicação mesmo durante esses picos. Por outro lado, convém definir uma capacidade máxima mais baixa que possa reduzir o throughput durante picos incomuns, mas que permita ao Neptune lidar com as workloads esperadas sem problemas, além de limitar os custos.

Configuração adicional para clusters e instâncias de banco de dados do Neptune Serverless

Além de definir a capacidade mínima e máxima para o cluster de banco de dados do Neptune Serverless, há algumas outras opções de configuração a serem consideradas.

Combinar instâncias sem servidor e provisionadas em um cluster de banco de dados

Um cluster de banco de dados não precisa ser apenas sem servidor. É possível criar uma combinação de instâncias sem servidor e provisionadas (uma configuração mista).

Por exemplo, suponha que você precise de mais capacidade de gravação do que está disponível em uma instância sem servidor. Nesse caso, é possível configurar o cluster com um gravador provisionado muito grande e ainda usar as instâncias sem servidor para os leitores.

Ou suponha que a workload de gravação no cluster varie, mas a workload de leitura permaneça estável. Nesse caso, você pode configurar o cluster com um gravador sem servidor e um ou mais leitores provisionados.

Consulte Usar o Amazon Neptune Serverless para obter informações sobre como criar um cluster de banco de dados de configuração mista.

Definir os níveis de promoção para instâncias do Neptune Serverless

No caso de clusters que contenham várias instâncias sem servidor ou uma mistura de instâncias sem servidor e provisionadas, preste atenção à configuração do nível de promoção de cada instância sem servidor. Essa configuração controla mais aspectos do comportamento das instâncias sem servidor do que das instâncias de banco de dados provisionadas.

No AWS Management Console, você especifica essa configuração usando a prioridade de failover em Configuração adicional nas páginas Criar banco de dados, Modificar instância e Adicionar leitor.

Você pode ver essa propriedade para instâncias existentes na coluna Nível de prioridade da página Bancos de dados. Você também pode ver essa propriedade na página de detalhes de uma instância ou um cluster de banco de dados.

No caso de instâncias provisionadas, a escolha do nível de 0 a 15 determina apenas a ordem na qual o Neptune escolhe qual instância de leitor promover ao gravador durante uma operação de failover. No caso de instâncias de leitor do Neptune Serverless, o número do nível também determina se a escala da instância será aumentada verticalmente para corresponder à capacidade da instância de gravador ou se será escalada independentemente dela, com base somente na própria workload.

As instâncias de leitor do Neptune Serverless no nível 0 ou 1 são mantidas em uma capacidade mínima pelo menos tão alta quanto a instância de gravador, para que estejam prontas para assumir o gravador em caso de failover. Caso o gravador seja uma instância provisionada, o Neptune estima a capacidade sem servidor equivalente e usa essa estimativa como a capacidade mínima para a instância de leitor sem servidor.

As instâncias de leitor do Neptune Serverless nos níveis 2 a 15 não têm a mesma restrição na capacidade mínima e são escaladas independentemente do gravador. Quando estiverem ociosas, elas poderão ter a escala reduzida verticalmente para o valor mínimo de NCU especificado no intervalo de capacidade do cluster. No entanto, isso poderá causar problemas se a workload de leitura aumentar rapidamente.

Manter a capacidade do leitor alinhada à capacidade do gravador

Um fator importante a ser lembrado é que convém garantir que as instâncias de leitor possam acompanhar a instância de gravador, para evitar atrasos excessivos na replicação. Isso é particularmente preocupante em duas situações em que as instâncias de leitor sem servidor não são escaladas automaticamente em sincronia com a instância de gravador:

- Quando o gravador é provisionado e os leitores são sem servidor.
- Quando o gravador é sem servidor e os leitores sem servidor estão nos níveis de promoção de 2 a 15.

Nos dois casos, defina a capacidade mínima sem servidor para corresponder à capacidade esperada do gravador, a fim de garantir que as operações do leitor não atinjam o tempo limite e causem reinicializações. No caso de uma instância de gravador provisionada, defina a capacidade mínima para corresponder à da instância provisionada. No caso de um gravador sem servidor, a configuração ideal pode ser mais difícil de prever.

Como o intervalo de capacidade da instância é definido em nível de cluster, todas as instâncias sem servidor são controladas pelas mesmas configurações de capacidade mínima e máxima. As instâncias de leitor nos níveis 0 e 1 são escaladas em sincronia com a instância de gravador, mas as instâncias nos níveis de promoção de 2 a 15 são escaladas independentemente umas das outras e da instância de gravador, dependendo da workload. Se você definir a capacidade mínima muito baixa, a escala das instâncias ociosas nos níveis de 2 a 15 poderá ser reduzida verticalmente de forma excessiva para ser aumentada com rapidez suficiente para lidar com uma explosão repentina na atividade do gravador.

Evitar definir um valor de tempo limite muito alto

Você poderá gerar custos inesperados se definir um valor de tempo limite de consulta muito alto em uma instância sem servidor.

Sem uma configuração de tempo limite razoável, você pode emitir acidentalmente uma consulta que exija um tipo de instância avançado e caro e que continue em execução por muito tempo, gerando custos nunca previstos. É possível evitar essa situação usando um valor de tempo limite que acomode a maioria das consultas e ocasione o término do tempo limite apenas para as excepcionalmente longas.

Isso vale tanto para valores de tempo limite de consulta gerais definidos com parâmetros quanto para valores de tempo limite por consulta definidos com dicas de consulta.

Otimizar a configuração do Neptune Serverless

Se o cluster de banco de dados do Neptune Serverless não estiver ajustado à workload em execução, você poderá perceber que ele não funciona de maneira ideal. É possível ajustar a configuração de capacidade mínima e/ou máxima para que ela possa ser escalada sem encontrar problemas de memória.

- Aumentar a configuração de capacidade mínima do cluster Isso pode corrigir a situação em que a instância ociosa é escalada para uma capacidade com menos memória do que a aplicação e os atributos habilitados precisam.
- Aumente a configuração de capacidade máxima do cluster. Isso pode corrigir a situação em que não é possível aumentar a escala de um banco de dados ocupado verticalmente até uma capacidade com memória suficiente para lidar com a workload e os atributos habilitados com uso intenso de memória.
- Altere a workload na instância em questão. Por exemplo, você pode adicionar instâncias de leitor ao cluster para distribuir a carga de leitura por mais instâncias.

- Ajuste as consultas da aplicação para que elas usem menos recursos.
- Tente usar uma instância provisionada que seja maior do que o máximo de NCUs disponíveis no Neptune Serverless, para ver se ela é mais adequada aos requisitos de memória e CPU da workload.

Usar o Amazon Neptune Serverless

É possível criar um cluster de banco de dados do Neptune como sem servidor ou, em alguns casos, converter um cluster de banco de dados existente para usar tecnologia sem servidor. Você também pode converter instâncias de banco de dados em um cluster de banco de dados sem servidor de e em instâncias sem servidor. Você só pode usar o Neptune Serverless em um dos Regiões da AWS locais onde ele é suportado, com algumas outras limitações (consulte). Restrições do Amazon Neptune Serverless

Você também pode usar a pilha do AWS CloudFormation do Neptune para criar um cluster de banco de dados do Neptune Serverless.

Criar um cluster de banco de dados que usa a tecnologia sem servidor

Para criar um cluster de banco de dados do Neptune que use a tecnologia sem servidor, faça-o usando o AWS Management Console da mesma forma que faz para criar um cluster provisionado. A diferença é que, no tamanho da instância de banco de dados, é necessário definir a classe da instância de banco de dados como sem servidor. Ao fazer isso, é necessário definir o intervalo de capacidade sem servidor para o cluster.

Você também pode criar um cluster de banco de dados sem servidor usando os comandos AWS CLI with como este (no Windows, substitua '\' por '^'):

```
aws neptune create-db-cluster \
  --region (an Região da AWS region that supports serverless) \
  --db-cluster-identifier (ID for the new serverless DB cluster) \
  --engine neptune \
  --engine-version (optional: 1.2.0.1 or above) \
  --serverless-v2-scaling-configuration "MinCapacity=1.0, MaxCapacity=128.0"
```

Também é possível especificar o parâmetro `serverless-v2-scaling-configuration` desta forma:

```
  --serverless-v2-scaling-configuration '{"MinCapacity":1.0, "MaxCapacity":128.0}'
```

Depois, você pode executar o comando `describe-db-clusters` para o atributo `ServerlessV2ScalingConfiguration`, que deve exibir as configurações de intervalo de capacidade especificadas:

```
"ServerlessV2ScalingConfiguration": {
    "MinCapacity": (the specified minimum number of NCUs),
    "MaxCapacity": (the specified maximum number of NCUs)
}
```

Converter uma instância ou um cluster de banco de dados existente em tecnologia sem servidor

Se você tiver um cluster de banco de dados do Neptune que esteja usando a versão 1.2.0.1 ou posterior do mecanismo, poderá convertê-lo em sem servidor. Esse processo gera algum tempo de inatividade.

A primeira etapa é adicionar um intervalo de capacidade ao cluster existente. Você pode fazer isso usando o AWS Management Console, ou usando um AWS CLI comando como este (no Windows, substitua '\' por '^'):

```
aws neptune modify-db-cluster \
  --db-cluster-identifier (your DB cluster ID) \
  --serverless-v2-scaling-configuration \
      MinCapacity=(minimum number of NCUs, such as  2.0), \
      MaxCapacity=(maximum number of NCUs, such as 24.0)
```

A próxima etapa é criar uma instância de banco de dados sem servidor para substituir a instância principal existente (o gravador) no cluster. Novamente, você pode fazer isso e todas as etapas subsequentes usando o AWS Management Console ou AWS CLI o. Em qualquer caso, especifique a classe da instância de banco de dados como sem servidor. O AWS CLI comando ficaria assim (no Windows, substitua '\' por '^'):

```
aws neptune create-db-instance \
  --db-instance-identifier (an instance ID for the new writer instance) \
  --db-cluster-identifier (ID of the DB cluster) \
  --db-instance-class db.serverless
```

```
  --engine neptune
```

Quando a nova instância de gravador estiver disponível, realize um failover para torná-la a instância de gravador do cluster:

```
aws neptune failover-db-cluster \
  --db-cluster-identifier (ID of the DB cluster) \
  --target-db-instance-identifier (instance ID of the new serverless instance)
```

Depois, exclua a instância de gravador antiga:

```
aws neptune delete-db-instance \
  --db-instance-identifier (instance ID of the old writer instance) \
  --skip-final-snapshot
```

Por fim, faça o mesmo para criar uma instância sem servidor para substituir cada instância de leitor provisionada existente que você gostaria de transformar em uma instância sem servidor e exclua as instâncias provisionadas existentes (nenhum failover é necessário para instâncias de leitor).

Modificar o intervalo de capacidade de um cluster de banco de dados sem servidor existente

É possível alterar o intervalo de capacidade de um cluster de banco de dados do Neptune Sem Servidor usando a AWS CLI da seguinte forma (no Windows, substitua "\" por "^"):

```
aws neptune modify-db-cluster \
  --region (an AWS region that supports serverless) \
  --db-cluster-identifier (ID of the serverless DB cluster) \
  --apply-immediately \
  --serverless-v2-scaling-configuration MinCapacity=4.0, MaxCapacity=32
```

A alteração do intervalo de capacidade causa alterações nos valores padrão de alguns parâmetros de configuração. O Neptune pode aplicar alguns desses novos padrões imediatamente, mas algumas das alterações de parâmetros dinâmicos são aplicadas somente após a reinicialização. O status `pending-reboot` indica que é necessária uma reinicialização para aplicar algumas alterações de parâmetro.

Alterar uma instância de banco de dados sem servidor para provisionada

Para converter uma instância do Neptune Sem Servidor em uma provisionada basta alterar a classe da instância para uma das classes de instâncias provisionadas. Consulte Modificar uma instância de banco de dados do Neptune (e aplicar imediatamente).

Monitorando a capacidade sem servidor com a Amazon CloudWatch

Você pode usá-lo CloudWatch para monitorar a capacidade e a utilização das instâncias sem servidor do Neptune em seu cluster de banco de dados. Há duas CloudWatch métricas que permitem monitorar a capacidade atual sem servidor, tanto no nível do cluster quanto no nível da instância:

- **`ServerlessDatabaseCapacity`**: como métrica em nível de instância, `ServerlessDatabaseCapacity` relata a capacidade atual da instância, em NCUs. Como métrica em nível de cluster, ela relata a média de todos os valores `ServerlessDatabaseCapacity` de todas as instâncias de banco de dados no cluster.
- **`NCUUtilization`**: essa métrica relata a porcentagem da capacidade possível que está sendo usada. Ela é calculada como a `ServerlessDatabaseCapacity` atual (na instância ou no cluster) dividida pela configuração de capacidade máxima do cluster de banco de dados.

  Se essa métrica se aproximar de 100% no cluster, o que significa que ele foi escalado para o valor alto possível, pense em aumentar a configuração de capacidade máxima.

  Se ela se aproximar de 100% para uma instância de leitor enquanto a instância de gravador não estiver próxima da capacidade máxima, pense em adicionar mais instâncias de leitor para distribuir a workload de leitura.

Observe que as métricas `CPUUtilization` e `FreeableMemory` têm significados um pouco diferentes para instâncias sem servidor e instâncias provisionadas. Em um contexto de tecnologia sem servidor, `CPUUtilization` é uma porcentagem calculada como a quantidade de CPU que está sendo usada atualmente dividida pela quantidade de CPU disponível na capacidade máxima. Da mesma forma, `FreeableMemory` relata a quantidade de memória que pode ser liberada e estaria disponível se uma instância estivesse na capacidade máxima.

O exemplo a seguir mostra como usar o AWS CLI no Linux para recuperar os valores de capacidade mínima, máxima e média de uma determinada instância de banco de dados, medidos a cada 10 minutos durante uma hora. O comando `date` do Linux especifica as horas de início e término em

relação à data e hora atuais. A função `sort_by` no parâmetro `--query` classifica os resultados cronologicamente com base no campo `Timestamp`:

```
aws cloudwatch get-metric-statistics \
  --metric-name "ServerlessDatabaseCapacity" \
  --start-time "$(date -d '1 hour ago')" \
  --end-time "$(date -d 'now')" \
  --period 600 \
  --namespace "AWS/Neptune"
  --statistics Minimum Maximum Average \
  --dimensions Name=DBInstanceIdentifier,Value=(instance ID) \
  --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' \
  --output table
```

Capturar alterações do grafo em tempo real usando os fluxos do Neptune

Os fluxos do Neptune registram em log todas as alterações feitas no grafo à medida que ocorrem e na ordem em que ocorrem, de forma totalmente gerenciada. Assim que você habilita os fluxos, o Neptune cuida da disponibilidade, do backup, da segurança e da validade.

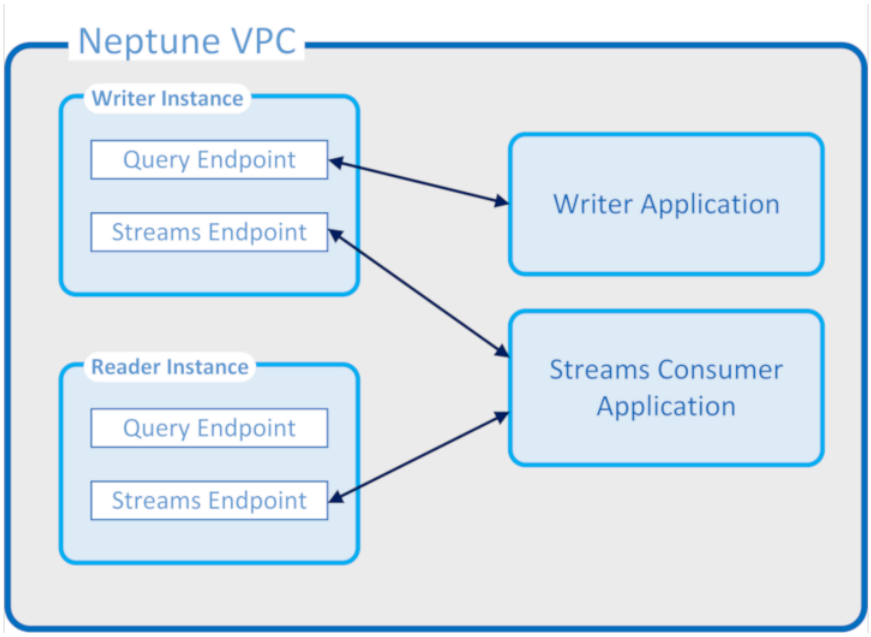
> **Note**
>
> Esse atributo estava disponível no modo de laboratório a partir de Versão 1.0.1.0.200463.0 (15/10/2019) e está disponível para uso em produção a partir da versão 1.0.2.2.R2 do mecanismo do Neptune.

Veja a seguir alguns dos muitos casos de uso em que você pode desejar capturar alterações em um gráfico à medida que ocorrem:

- Você pode desejar que seu aplicativo notifique as pessoas automaticamente quando determinadas alterações forem feitas.
- Talvez você também queira manter uma versão atual dos seus dados gráficos em outro armazenamento de dados, como Amazon OpenSearch Service, Amazon ou Amazon ElastiCache Simple Storage Service (Amazon S3).

O Neptune usa o mesmo armazenamento nativo para o fluxo de logs de alterações e para os dados do grafo. Ele grava entradas de log de alterações de forma síncrona junto com a transação que fez essas alterações. Você recupera esses registros de alterações do fluxo de log usando uma API REST HTTP. (Para obter informações, consulte Chamar a API do Streams.)

O diagrama a seguir mostra como os dados do log de alterações podem ser recuperados dos fluxos do Neptune.



Garantias dos fluxos do Neptune

- As alterações feitas por uma transação ficam imediatamente disponíveis para leitura do gravador e dos leitores assim que a transação é concluída (além de qualquer atraso de replicação normal nos leitores).
- Os registros de alterações aparecem estritamente de forma sequencial, na ordem em que ocorreram (isso inclui as alterações feitas em uma transação).
- Os fluxos de alterações não contêm duplicatas. Cada alteração é registrada em log somente uma vez.
- Os fluxos de alterações são completos. Nenhuma alteração é perdida ou omitida.
- Os fluxos de alterações contêm todas as informações necessárias para determinar o estado completo do próprio banco de dados a qualquer momento, desde que o estado inicial seja conhecido.
- O Streams pode ser ativado ou desativado a qualquer momento.

Propriedades operacionais de fluxos do Neptune

- O fluxo de logs de alterações é totalmente gerenciado.
- Os dados do log de alterações são gravados de forma síncrona como parte da mesma transação que faz uma alteração.

- Quando os fluxos do Neptune estão habilitados, são geradas cobranças de E/S e armazenamento associadas aos dados do log de alterações.
- Por padrão, os registros de alterações são removidos automaticamente uma semana após a criação. A partir da versão 1.2.0.0 do mecanismo, esse período de retenção pode ser alterado usando o parâmetro de cluster de banco de dados neptune_streams_expiry_days para qualquer número de dias entre um e noventa.
- O desempenho de leitura nos fluxos é dimensionado junto com as instâncias.
- Você pode obter alta disponibilidade e taxa de transferência de leitura usando réplicas de leitura. Não há limite para o número de leitores de fluxos que você pode criar e usar simultaneamente.
- Os dados do log de alterações são replicados em várias zonas de disponibilidade, o que os torna resilientes.
- Os dados do log são tão seguros quanto os dados do próprio gráfico. Os dados podem ser criptografados em repouso e em trânsito. O acesso pode ser controlado usando IAM, Amazon VPC e AWS Key Management Service ()AWS KMS. Assim como os dados do gráfico, eles podem ser copiados e restaurados posteriormente usando point-in-time restaurações (PITR).
- A gravação síncrona dos dados de fluxo como parte de cada transação provoca uma leve degradação no desempenho geral de gravação.
- Os dados do fluxo não são fragmentados, porque o Neptune é um único fragmento por design.
- A API `GetRecords` do fluxo de logs usa os mesmos recursos que todas as outras operações de grafo do Neptune. Isso significa que os clientes precisam balancear a carga entre solicitações de fluxo e outras solicitações de banco de dados.
- Quando os fluxos são desabilitados, todos os dados do log se tornam inacessíveis imediatamente. Isso significa que você deve ler todos os dados de seu interesse antes de desabilitar o registro em log.
- Atualmente, não há integração nativa com AWS Lambda o. O fluxo de logs não gera um evento que possa acionar uma função do Lambda.

Tópicos

- Usar a replicação entre regiões de fluxos do Neptune para recuperação de desastres

Usar os fluxos do Neptune

Com o atributo de fluxos do Neptune, é possível gerar uma sequência completa de entradas no log de alterações que registram todas as alterações feitas em dados de grafos à medida que elas ocorrem. Para obter uma visão geral desse recurso, consulte Capturar alterações do grafo em tempo real usando os fluxos do Neptune.

Tópicos

- Habilitar os fluxos do Neptune
- Desabilitar os fluxos do Neptune
- Chamar a API REST de fluxos do Neptune
- Formato de resposta da API de fluxos do Neptune
- Exceções da API de fluxos do Neptune

Habilitar os fluxos do Neptune

É possível habilitar ou desabilitar os fluxos do Neptune a qualquer momento ao configurar o parâmetro do cluster de banco de dados do `neptune_streams`. Definir o parâmetro como 1 habilita o Streams e defini-lo como `0` desabilita o Streams.

> **Note**
>
> Depois de alterar o parâmetro de cluster de banco de dados `neptune_streams`, é necessário reinicializar todas as instâncias de bancos de dados no cluster para que a alteração tenha efeito.

É possível definir o parâmetro do cluster de banco de dados neptune_streams_expiry_days para controlar quantos dias, de um a noventa, esses registros de fluxo permanecem no servidor antes de serem excluídos. O padrão é 7.

Os fluxos do Neptune foram inicialmente introduzidos como um atributo experimental que você habilitou ou desabilitou no modo de laboratório usando o parâmetro `neptune_lab_mode` de cluster

de banco de dados (consulte Modo de laboratório do Neptune). O uso do modo de laboratório para habilitar o Streams está defasado e será desativado no futuro.

Desabilitar os fluxos do Neptune

Você pode desativar os fluxos do Neptune a qualquer momento em que ele estiver em execução.

Para desativar o Streams, atualize o grupo de parâmetros do cluster de banco de dados para que o valor do parâmetro `neptune_streams` seja definido como 0.

> **Important**
>
> Assim que o Streams for desativado, você não poderá mais acessar os dados do log de alterações. Leia o que é de seu interesse antes de desativar o Streams.

Chamar a API REST de fluxos do Neptune

Você acessa os fluxos do Neptune usando uma API REST que envia uma solicitação GET HTTP a um dos seguintes endpoints locais:

- Para um banco de dados gráfico do SPARQL: `https://`*`Neptune-DNS`*`:8182/sparql/stream`.
- Para um banco de dados de grafos do Gremlin ou do openCypher: `https://`*`Neptune-DNS`*`:8182/propertygraph/stream` ou `https://`*`Neptune-DNS`*`:8182/pg/stream`.

> **Note**
>
> A partir da versão 1.1.0.0 do mecanismo, o endpoint de fluxos do Gremlin (`https://`*`Neptune-DNS`*`:8182/gremlin/stream`) está sendo descontinuado, junto com o formato de saída associado (`GREMLIN_JSON`). Ele ainda é compatível com versões anteriores, mas pode ser removido em versões futuras.

Somente uma operação HTTP `GET` é permitida.

O Neptune é compatível com a compactação `gzip` da resposta, desde que a solicitação HTTP inclua um cabeçalho `Accept-Encoding` que especifique `gzip` como um formato de compactação aceito (ou seja, `"Accept-Encoding: gzip"`).

Parâmetros

- `limit`: longo, opcional. Intervalo: de um a cem mil. Padrão: 10.

  Especifica o número máximo de registros a serem retornados. Há também um limite de tamanho de 10 MB na resposta que não pode ser modificado e que tem precedência sobre o número de registros especificado no parâmetro `limit`. A resposta incluirá um registro de violação de limite se o limite de 10 MB tiver sido atingido.

- `iteratorType`: string, opcional.

  Esse parâmetro pode ter um dos valores a seguir:

  - `AT_SEQUENCE_NUMBER`(padrão): indica que a leitura deve começar a partir do número de sequência de eventos especificado em conjunto pelos parâmetros `commitNum` e `opNum`.
  - `AFTER_SEQUENCE_NUMBER`: indica que a leitura deve começar logo após o número de sequência de eventos especificado em conjunto pelos parâmetros `commitNum` e `opNum`.
  - `TRIM_HORIZON`: indica que a leitura deve começar no último registro não truncado no sistema, que é o registro mais antigo não expirado (ainda não excluído) no fluxo de logs de alterações. Esse modo é útil durante a inicialização do aplicativo, quando você não tem um número inicial específico de sequência de eventos.
  - `LATEST`: indica que a leitura deve começar no registro mais recente no sistema, que é o registro mais recente não expirado (ainda não excluído) no fluxo de logs de alterações. Isso é útil quando há a necessidade de ler registros da parte superior atual dos fluxos para não processar registros antigos, como durante a recuperação de desastres ou uma atualização sem tempo de inatividade. Observe que, nesse modo, há no máximo apenas um registro exibido.

- `commitNum`: longo, obrigatório quando iteratorType é `AT_SEQUENCE_NUMBER` ou `AFTER_SEQUENCE_NUMBER`.

  O número de confirmação do registro inicial a ser lido no fluxo do log de alterações.

  Este parâmetro é ignorado quando `iteratorType` é `TRIM_HORIZON` ou `LATEST`.

- `opNum`: longo, opcional (o padrão é 1).

  O número de sequência da operação dentro da confirmação especificada da qual começar a ler nos dados do fluxo do log de alterações.

As operações que alteram os dados do gráfico do SPARQL geralmente geram apenas um único registro de alteração por operação. No entanto, as operações que alteram dados de gráficos do Gremlin podem gerar vários registros de alterações por operação, como nos exemplos a seguir:

- `INSERT`: um vértice do Gremlin pode ter vários rótulos, e um elemento do Gremlin pode ter várias propriedades. Um registro de alteração separado é gerado para cada rótulo e propriedade quando um elemento é inserido.
- `UPDATE`: quando uma propriedade de elemento do Gremlin é alterada, dois registros de alteração são gerados: o primeiro para remover o valor anterior e o segundo para inserir o novo valor.
- `DELETE`: um registro de alteração separado é gerado para cada propriedade de elemento excluída. Por exemplo, quando uma borda do Gremlin com propriedades é excluída, um registro de alteração é gerado para cada uma das propriedades e, depois disso, é gerado um para a exclusão do rótulo da borda.

  Quando um vértice do Gremlin é excluído, todas as propriedades de borda de entrada e saída são excluídas primeiro, depois os rótulos da borda, depois as propriedades do vértice e, por fim, os rótulos do vértice. Cada uma dessas exclusões gera um registro de alteração.

Formato de resposta da API de fluxos do Neptune

Uma resposta a uma solicitação de API REST dos fluxos do Neptune tem os seguintes campos:

- `lastEventId`: identificador da sequência da última alteração na resposta do fluxo. Um ID de evento é composto de dois campos: um `commitNum` identifica uma transação que alterou o gráfico, e um `opNum` identifica uma operação específica dentro dessa transação. Isso é mostrado no exemplo a seguir.

  ```
  "eventId": {
    "commitNum": 12,
    "opNum": 1
  }
  ```

- `lastTrxTimestamp`: a hora em que a confirmação da transação foi solicitada, em milissegundos a partir da época do Unix.
- `format`: o formato da serialização dos registros de alterações que estão sendo gerados. Os valores possíveis são `PG_JSON` para registros de alterações do Gremlin ou do openCypher, `NQUADS` para registros de alterações do SPARQL.

- `records`: uma matriz de registros serializados do fluxo de logs de alterações incluídos na resposta. Cada registro na matriz `records` contém os seguintes campos:
  - `commitTimestamp`: a hora em que a confirmação da transação foi solicitada, em milissegundos a partir da época do Unix.
  - `eventId`: o identificador da sequência da registro de alteração do fluxo.
  - `data`— O registro serializado de Gremlin, SPARQL ou alteração. OpenCypher Os formatos de serialização de cada registro são descritos em mais detalhes na próxima seção, Formatos de serialização nos fluxos do Neptune.
  - `op`: a operação que criou a alteração.
  - `isLastOp`: presente somente se essa operação for a última da transação. Quando presente, está definido como `true`. Útil para garantir que uma transação inteira seja consumida.
- `totalRecords`: o número total de registros na resposta.

Por exemplo, a seguinte resposta exibe dados de alteração do Gremlin para uma transação que contém mais de uma operação:

```
{
  "lastEventId": {
    "commitNum": 12,
    "opNum": 1
  },
  "lastTrxTimestamp": 1560011610678,
  "format": "PG_JSON",
  "records": [
    {
      "commitTimestamp": 1560011610678,
      "eventId": {
        "commitNum": 1,
        "opNum": 1
      },
      "data": {
        "id": "d2b59bf8-0d0f-218b-f68b-2aa7b0b1904a",
        "type": "vl",
        "key": "label",
        "value": {
          "value": "vertex",
          "dataType": "String"
        }
      },
```

```
      "op": "ADD"
    }
  ],
  "totalRecords": 1
}
```

A resposta a seguir gera dados de alteração do SPARQL para a última operação em uma transação (a operação identificada por `EventId(97, 1)` na transação número 97).

```
{
  "lastEventId": {
    "commitNum": 97,
    "opNum": 1
  },
  "lastTrxTimestamp": 1561489355102,
  "format": "NQUADS",
  "records": [
    {
      "commitTimestamp": 1561489355102,
      "eventId": {
        "commitNum": 97,
        "opNum": 1
      },
      "data": {
        "stmt": "<https://test.com/s> <https://test.com/p> <https://test.com/o> .\n"
      },
      "op": "ADD",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

Exceções da API de fluxos do Neptune

A tabela a seguir descreve as exceções dos fluxos do Neptune.

Código de erro	Código HTTP	OK para repetir?	Message
`InvalidParameterException`	400	Não	Um out-of-range valor ou inválido

Código de erro	Código HTTP	OK para repetir?	Message
			foi fornecido como parâmetro de entrada.
`ExpiredStreamException`	400	Não	Todos os registros solicitados excedem a idade máxima permitida e expiraram.
`ThrottlingException`	500	Sim	A taxa de solicitações excede a taxa de transferência máxima.
`StreamRecordsNotFoundException`	404	Não	Não foi possível encontrar o recurso solicitado. O fluxo pode não ter sido especificado corretamente.
`MemoryLimitExceededException`	500	Sim	O processamento da solicitação não foi bem-sucedido devido à falta de memória, mas pode ser repetido quando o servidor estiver menos ocupado.

Formatos de serialização nos fluxos do Neptune

O Amazon Neptune usa dois formatos diferentes para serializar dados de alterações nos grafos para fluxos de log, dependendo de se o grafo foi criado usando o Gremlin ou o SPARQL.

Os dois formatos compartilham um formato comum de serialização de registros, conforme descrito em Formato de resposta da API de fluxos do Neptune, que contém os seguintes campos:

- `commitTimestamp`: a hora em que a confirmação da transação foi solicitada, em milissegundos a partir da época do Unix.
- `eventId`: o identificador da sequência da registro de alteração do fluxo.
- `data`— O registro serializado de Gremlin, SPARQL ou alteração. OpenCypher Os formatos de serialização de cada registro são descritos em mais detalhes na próxima seção.
- `op`: a operação que criou a alteração.

Tópicos

Formato da serialização de alterações do PG_JSON

> **Note**
>
> A partir da versão 1.1.0.0 do mecanismo, o formato de saída de fluxos do Gremlin (`GREMLIN_JSON`) produzido pelo endpoint de fluxos do Gremlin (`https://Neptune-DNS:8182/gremlin/stream`) está sendo descontinuado. Ele foi substituído pelo PG_JSON, que atualmente é idêntico a `GREMLIN_JSON`.

Um registro de alterações do Gremlin ou do openCypher, contido no campo `data` de uma resposta do fluxo de logs, tem os seguintes campos:

- `id`: string, obrigatório.

  O ID do elemento do Gremlin ou do openCypher.
- `type`: string, obrigatório.

  O tipo desse elemento do Gremlin ou do openCypher Deve ser um dos seguintes:
  - `vl`: rótulo de vértice para Gremlin; rótulo de nó para openCypher.
  - `vp`: propriedades de vértice para Gremlin; propriedades de nós para openCypher.
  - `e`: rótulo de borda e borda para Gremlin; relacionamento e tipo de relacionamento para openCypher.
  - `ep`: propriedades de borda para Gremlin; propriedades de relacionamento para openCypher.

- `key`: string, obrigatório.

  O nome da propriedade. Para rótulos de elementos, o nome é "label".

- `value`: objeto `value`, obrigatório.

  Trata-se de um objeto JSON que contém um campo `value` para o próprio valor, e um campo `datatype` para o tipo de dados JSON desse valor.

```
  "value": {
    "value": "the new value",
    "dataType": "the JSON datatype of the new value"
  }
```

- `from`: string, opcional.

  Se esta for uma borda (tipo = “e”), o ID do vértice de correspondente ou do nó de origem.

- `to`: string, opcional.

  Se esta for uma borda (tipo = “e”), o ID do vértice para correspondente ou nó de destino.

Exemplos do Gremlin

- Veja a seguir um exemplo de um rótulo de vértice do Gremlin.

```
{
  "id": "an ID string",
  "type": "vl",
  "key": "label",
  "value": {
    "value": "the new value of the vertex label",
    "dataType": "String"
  }
}
```

- Veja a seguir um exemplo de uma propriedade de vértice do Gremlin.

```
{
  "id": "an ID string",
  "type": "vp",
  "key": "the property name",
  "value": {
```

```
    "value": "the new value of the vertex property",
    "dataType": "the datatype of the vertex property"
  }
}
```

- Veja a seguir um exemplo de uma borda padrão do Gremlin.

```
{
  "id": "an ID string",
  "type": "e",
  "key": "label",
  "value": {
    "value": "the new value of the edge",
    "dataType": "String"
  },
  "from": "the ID of the corresponding "from" vertex",
  "to": "the ID of the corresponding "to" vertex"
}
```

Exemplos do openCypher

- Veja um exemplo de rótulo de nó do openCypher.

```
{
  "id": "an ID string",
  "type": "vl",
  "key": "label",
  "value": {
    "value": "the new value of the node label",
    "dataType": "String"
  }
}
```

- Veja um exemplo de propriedade de nó do openCypher.

```
{
  "id": "an ID string",
  "type": "vp",
  "key": "the property name",
  "value": {
    "value": "the new value of the node property",
    "dataType": "the datatype of the node property"
```

```
  }
}
```

- Veja um exemplo de relacionamento do openCypher.

```
{
  "id": "an ID string",
  "type": "e",
  "key": "label",
  "value": {
    "value": "the new value of the relationship",
    "dataType": "String"
  },
  "from": "the ID of the corresponding source node",
  "to": "the ID of the corresponding target node"
}
```

Formato de serialização de alterações de NQUADS do SPARQL

O Neptune registra alterações nos quadrantes do SPARQL no grafo usando a linguagem Resource Description Framework (RDF) `N-QUADS` definida na especificação W3C RDF 1.1 N-Quads.

O campo `data` no registro de alterações simplesmente contém um campo `stmt` que contém uma instrução N-QUADS que expressa o quad alterado, como no exemplo a seguir.

```
"stmt" : "<https://test.com/s> <https://test.com/p> <https://test.com/o> .\n"
```

Exemplos de fluxo do Neptune

Os exemplos a seguir mostram como acessar dados de fluxo de logs de alterações no Amazon Neptune.

Tópicos

- Log de alterações AT_SEQUENCE_NUMBER
- Log de alterações AFTER_SEQUENCE_NUMBER
- Log de alterações TRIM_HORIZON
- Log de alterações LATEST
- Log de alterações de compactação

Log de alterações **AT_SEQUENCE_NUMBER**

O exemplo a seguir mostra um log de alterações AT_SEQUENCE_NUMBER do Gremlin ou do openCypher.

```
curl -s "https://Neptune-DNS:8182/propertygraph/stream?
limit=1&commitNum=1&opNum=1&iteratorType=AT_SEQUENCE_NUMBER" |jq
{
  "lastEventId": {
    "commitNum": 1,
    "opNum": 1
  },
  "lastTrxTimestamp": 1560011610678,
  "format": "PG_JSON",
  "records": [
    {
      "eventId": {
        "commitNum": 1,
        "opNum": 1
      },
      "commitTimestamp": 1560011610678,
      "data": {
        "id": "d2b59bf8-0d0f-218b-f68b-2aa7b0b1904a",
        "type": "vl",
        "key": "label",
        "value": {
          "value": "vertex",
          "dataType": "String"
        }
      },
      "op": "ADD",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

Este mostra um exemplo do SPARQL de um log de alterações AT_SEQUENCE_NUMBER.

```
curl -s "https://localhost:8182/sparql/stream?
limit=1&commitNum=1&opNum=1&iteratorType=AT_SEQUENCE_NUMBER" |jq
{
  "lastEventId": {
```

```
    "commitNum": 1,
    "opNum": 1
  },
  "lastTrxTimestamp": 1571252030566,
  "format": "NQUADS",
  "records": [
    {
      "eventId": {
        "commitNum": 1,
        "opNum": 1
      },
      "commitTimestamp": 1571252030566,
      "data": {
        "stmt": "<https://test.com/s> <https://test.com/p> <https://test.com/o> .\n"
      },
      "op": "ADD",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

Log de alterações **AFTER_SEQUENCE_NUMBER**

O exemplo a seguir mostra um log de alterações AFTER_SEQUENCE_NUMBER do Gremlin ou do openCypher.

```
curl -s "https://Neptune-DNS:8182/propertygraph/stream?
limit=1&commitNum=1&opNum=1&iteratorType=AFTER_SEQUENCE_NUMBER" |jq
{
  "lastEventId": {
    "commitNum": 2,
    "opNum": 1
  },
  "lastTrxTimestamp": 1560011633768,
  "format": "PG_JSON",
  "records": [
    {
      "commitTimestamp": 1560011633768,
      "eventId": {
        "commitNum": 2,
        "opNum": 1
      },
```

```
      "data": {
        "id": "d2b59bf8-0d0f-218b-f68b-2aa7b0b1904a",
        "type": "vl",
        "key": "label",
        "value": {
          "value": "vertex",
          "dataType": "String"
        }
      },
      "op": "REMOVE",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

Log de alterações **TRIM_HORIZON**

O exemplo a seguir mostra um log de alterações TRIM_HORIZON do Gremlin ou do openCypher.

```
curl -s "https://Neptune-DNS:8182/propertygraph/stream?
limit=1&iteratorType=TRIM_HORIZON" |jq
{
  "lastEventId": {
    "commitNum": 1,
    "opNum": 1
  },
  "lastTrxTimestamp": 1560011610678,
  "format": "PG_JSON",
  "records": [
    {
      "commitTimestamp": 1560011610678,
      "eventId": {
        "commitNum": 1,
        "opNum": 1
      },
      "data": {
        "id": "d2b59bf8-0d0f-218b-f68b-2aa7b0b1904a",
        "type": "vl",
        "key": "label",
        "value": {
          "value": "vertex",
          "dataType": "String"
```

```
        }
      },
      "op": "ADD",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

Log de alterações **LATEST**

O exemplo a seguir mostra um log de alterações LATEST do Gremlin ou do openCypher. Observe que os parâmetros da API `limit`, `commitNum` e opNum são totalmente opcionais.

```
curl -s "https://Neptune-DNS:8182/propertygraph/stream?iteratorType=LATEST" | jq
{
  "lastEventId": {
    "commitNum": 21,
    "opNum": 4
  },
  "lastTrxTimestamp": 1634710497743,
  "format": "PG_JSON",
  "records": [
    {
      "commitTimestamp": 1634710497743,
      "eventId": {
        "commitNum": 21,
        "opNum": 4
      },
      "data": {
        "id": "24be4e2b-53b9-b195-56ba-3f48fa2b60ac",
        "type": "e",
        "key": "label",
        "value": {
          "value": "created",
          "dataType": "String"
        },
        "from": "4",
        "to": "5"
      },
      "op": "REMOVE",
      "isLastOp": true
    }
```

```
  ],
  "totalRecords": 1
}
```

Log de alterações de compactação

O exemplo a seguir mostra um log de alterações de compactação do Gremlin ou do openCypher.

```
curl -sH \
  "Accept-Encoding: gzip" \
  "https://Neptune-DNS:8182/propertygraph/stream?limit=1&commitNum=1" \
  -H "Accept-Encoding: gzip" \
  -v |gunzip -|jq
> GET /propertygraph/stream?limit=1 HTTP/1.1
> Host: localhost:8182
> User-Agent: curl/7.64.0
> Accept: /
> Accept-Encoding: gzip
*> Accept-Encoding: gzip*
>
< HTTP/1.1 200 OK
< Content-Type: application/json; charset=UTF-8
< Connection: keep-alive
*< content-encoding: gzip*
< content-length: 191
<
{ [191 bytes data]
Connection #0 to host localhost left intact
{
  "lastEventId": "1:1",
  "lastTrxTimestamp": 1558942160603,
  "format": "PG_JSON",
  "records": [
    {
      "commitTimestamp": 1558942160603,
      "eventId": "1:1",
      "data": {
        "id": "v1",
        "type": "vl",
        "key": "label",
        "value": {
          "value": "person",
          "dataType": "String"
```

```
        }
      },
      "op": "ADD",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

Usando AWS CloudFormation para configurar a replicação de Neptune-para-Neptune com o aplicativo Streams Consumer

Você pode usar um AWS CloudFormation modelo para configurar o aplicativo consumidor do Neptune Streams para oferecer suporte à replicação de Netuno para Netuno.

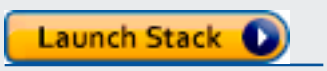
Tópicos

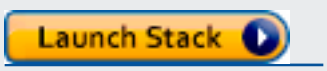
- Escolha um AWS CloudFormation modelo para sua região
- Adicionar detalhes sobre a pilha de consumidor de fluxos do Neptune que você está criando
- Execute o AWS CloudFormation modelo
- Como atualizar o instrumento de sondagem de fluxos com os artefatos mais recentes do Lambda

Escolha um AWS CloudFormation modelo para sua região

Para iniciar a AWS CloudFormation pilha apropriada no AWS CloudFormation console, escolha um dos botões Iniciar pilha na tabela a seguir, dependendo da AWS região que você deseja usar.

Região	Visualização	Visualizar no Designer	Executar
Leste dos EUA (Norte da Virgínia)	Visão	Visualizar no Designer	Launch Stack
Leste dos EUA (Ohio)	Visão	Visualizar no Designer	Launch Stack
Oeste dos EUA (N. da Califórnia)	Visão	Visualizar no Designer	Launch Stack

Região	Visualização	Visualizar no Designer	Executar
Oeste dos EUA (Oregon)	Visão	Visualizar no Designer	
Canadá (Central)	Visão	Visualizar no Designer	
América do Sul (São Paulo)	Visão	Visualizar no Designer	
Europa (Estocolmo)	Visão	Visualizar no Designer	
Europa (Irlanda)	Visão	Visualizar no Designer	
Europa (Londres)	Visão	Visualizar no Designer	
Europa (Paris)	Visão	Visualizar no Designer	
Europa (Frankfurt)	Visão	Visualizar no Designer	
Oriente Médio (Barém)	Visão	Visualizar no Designer	
Oriente Médio (Emirados Árabes Unidos)	Visão	Visualizar no Designer	
Israel (Tel Aviv)	Visão	Visualizar no Designer	
África (Cidade do Cabo)	Visão	Visualizar no Designer	

Região	Visualização	Visualizar no Designer	Executar
Ásia-Pacífico (Tóquio)	Visão	Visualizar no Designer	
Ásia-Pacífico (Hong Kong)	Visão	Visualizar no Designer	
Ásia-Pacífico (Seul)	Visão	Visualizar no Designer	
Ásia-Pacífico (Singapura)	Visão	Visualizar no Designer	
Ásia-Pacífico (Sydney)	Visão	Visualizar no Designer	
Ásia-Pacífico (Mumbai)	Visão	Visualizar no Designer	
China (Pequim)	Visão	Visualizar no Designer	
China (Ningxia)	Visão	Visualizar no Designer	
AWS GovCloud (Oeste dos EUA)	Visão	Visualizar no Designer	
AWS GovCloud (Leste dos EUA)	Visão	Visualizar no Designer	

Na página Create Stack (Criar pilha), selecione Next (Avançar).

Adicionar detalhes sobre a pilha de consumidor de fluxos do Neptune que você está criando

A página Specify Stack Details (Especificar detalhes da pilha) fornece propriedades e parâmetros que você pode usar para controlar a configuração do aplicativo:

Nome da pilha — O nome da nova AWS CloudFormation pilha que você está criando. Geralmente, você pode usar o valor padrão, `NeptuneStreamPoller`.

Em Parameters (Parâmetros), forneça o seguinte:

Configuração de rede para a VPC na qual o consumidor de fluxos é executado

- **`VPC`**: forneça o nome da VPC na qual a função do Lambda de sondagem será executada.
- **`SubnetIDs`**: as sub-redes para as quais será estabelecida uma interface de rede. Adicione sub-redes correspondentes ao cluster do Neptune.
- **`SecurityGroupIds`**: forneça os IDs dos grupos de segurança que concedem o acesso de entrada de gravação ao cluster de banco de dados de origem do Neptune.
- **`RouteTableIds`**: isso é necessário para criar um endpoint do Amazon DynamoDB na VPC do Neptune, caso ainda não tenha um. Forneça uma lista separada por vírgulas de IDs de tabela de rotas associados às sub-redes.
- **`CreateDDBVPCEndPoint`**: um valor booliano cujo padrão é `true`, indicando se é necessário ou não criar um endpoint da VPC de banco de dados do Dynamo. Você só precisará alterá-lo para `false` se já tiver criado um endpoint do DynamoDB na VPC.
- **`CreateMonitoringEndPoint`**: um valor booliano cujo padrão é `true`, indicando se é necessário ou não criar um endpoint da VPC de monitoramento. Você só precisará alterá-lo para `false` se já tiver criado um endpoint de monitoramento na VPC.

Instrumento de sondagem de fluxos

- **`ApplicationName`**: geralmente, você pode manter a configuração padrão (`NeptuneStream`). Se você usar um nome diferente, ele deverá ser exclusivo.
- **`LambdaMemorySize`**: usado para definir o tamanho da memória disponível para a função do instrumento de sondagem do Lambda. O valor padrão é de 2.048 megabytes.
- **`LambdaRuntime`**: a linguagem usada na função do Lambda que recupera os itens do fluxo do Neptune. É possível definir como `python3.9` ou `java8`.

- **`LambdaS3Bucket`**: o bucket do Amazon S3 que contém artefatos de código do Lambda. Deixe em branco, a menos que esteja usando uma função de sondagem personalizada do Lambda, carregada em outro bucket do Amazon S3.
- **`LambdaS3Key`**: a chave do Amazon S3 que corresponde aos seus artefatos de código do Lambda. Deixe em branco, a menos que esteja usando uma função de sondagem personalizada do Lambda.
- **`LambdaLoggingLevel`**: em geral, mantenha a configuração padrão, que é `INFO`.
- **`ManagedPolicies`**: lista as políticas gerenciadas a serem usadas na execução da função do Lambda. Em geral, deixe em branco, a menos que esteja usando uma função de sondagem personalizada do Lambda.
- **`StreamRecordsHandler`**: em geral, deixe em branco, a menos que esteja usando um manipulador personalizado para os registros nos fluxos do Neptune.
- **`StreamRecordsBatchSize`**: o número máximo de registros a serem obtidos no fluxo. É possível usar esse parâmetro para ajustar o desempenho. Recomendamos iniciar com o valor padrão (`5000`). O máximo permitido é 10.000. Quanto maior o número, menor serão as chamadas de rede necessárias para ler os registros do fluxo, mas maior será a memória necessária para processar os registros. Valores mais baixos desse parâmetro geram uma menor taxa de transferência.
- **`MaxPollingWaitTime`**: o tempo máximo de espera entre duas pesquisas (em segundos). Determina com que frequência o instrumento de sondagem do Lambda será invocado para sondar os fluxos do Neptune. Defina esse valor como 0 para a sondagem contínua. O valor máximo é de 3.600 segundos (1 hora). Recomendamos o valor padrão (60 segundos) para começar, dependendo da rapidez com que os dados do gráfico mudam.
- **`MaxPollingInterval`**: o período máximo de sondagem contínua (em segundos). Use isso para definir um tempo limite para a função de sondagem do Lambda. O valor deve estar no intervalo entre cinco segundos e novecentos segundos. Recomendamos começar com o valor padrão (600 segundos).
- **`StepFunctionFallbackPeriod`**— O número de unidades de step-function-fallback-period espera pelo poller, após o qual a função step é chamada por meio do Amazon CloudWatch Events para se recuperar de uma falha. Recomendamos iniciar com o valor padrão (5 minutos).
- **`StepFunctionFallbackPeriodUnit`**: as unidades de tempo usadas para medir o `StepFunctionFallbackPeriodUnit` anterior (`minutes`, `hours` ou `days`). Em geral, o padrão (`minutes`) é suficiente.

Fluxo do Neptune

- **`NeptuneStreamEndpoint`**: (obrigatório) o endpoint do fluxo de origem do Neptune. Isso assume uma destas duas formas:
  - **`https://`*`your DB cluster`*`:`*`port`*`/propertygraph/stream`** (ou o alias, `https://`*`your DB cluster`*`:`*`port`*`/pg/stream`).
  - **`https://`*`your DB cluster`*`:`*`port`*`/sparql/stream`**.
- **`Neptune Query Engine`**: escolha Gremlin, openCypher ou SPARQL.
- **`IAMAuthEnabledOnSourceStream`**: se o cluster de banco de dados do Neptune estiver usando a autenticação do IAM, defina esse parâmetro como `true`.
- **`StreamDBClusterResourceId`**: se o cluster de banco de dados do Neptune estiver usando a autenticação do IAM, defina esse parâmetro como o ID do recurso do cluster. O ID do recurso não é igual ao ID do cluster. Em vez disso, o formato é: `cluster-` seguido por 28 caracteres alfanuméricos. Ele pode ser encontrado em Detalhes do cluster no console do Neptune.

Cluster de banco de dados do Neptune de destino

- **`TargetNeptuneClusterEndpoint`**: o endpoint do cluster (somente nome do host) do cluster de backup de destino.

  Observe que, se você especificar `TargetNeptuneClusterEndpoint`, não poderá especificar `TargetSPARQLUpdateEndpoint`.
- **`TargetNeptuneClusterPort`**: o número da porta do cluster de destino.

  Observe que, se você especificar `TargetSPARQLUpdateEndpoint`, a configuração de `TargetNeptuneClusterPort` será ignorada.
- **`IAMAuthEnabledOnTargetCluster`**: defina como verdadeiro se a autenticação do IAM precisar ser habilitada no cluster de destino.
- **`TargetAWSRegion`**— A AWS região do cluster de backup de destino, como`us-east-1`). Você deve fornecer esse parâmetro somente quando a AWS região do cluster de backup de destino for diferente da região do cluster de origem do Neptune, como no caso da replicação entre regiões. Se as regiões de origem e de destino forem iguais, esse parâmetro será opcional.

  Observe que, se o `TargetAWSRegion` valor não for uma AWS região válida compatível com o Neptune, o processo falhará.

- **`TargetNeptuneDBClusterResourceId`**: opcional: isso só é necessário quando a autenticação do IAM está habilitada no cluster de banco de dados de destino. Defina como o ID do recurso do cluster de destino.
- **`SPARQLTripleOnlyMode`**: sinalizador booliano que determina se o modo somente triplo está habilitado. No modo somente triplo, não há replicação de grafo nomeado. O valor padrão é `false`.
- **`TargetSPARQLUpdateEndpoint`**: URL do endpoint de destino para atualização do SPARQL, como `https://abc.com/xyz`. Esse endpoint pode ser qualquer armazenamento SPARQL compatível com quádruplos ou triplos.

  Observe que, se você especificar `TargetSPARQLUpdateEndpoint`, também não poderá especificar `TargetNeptuneClusterEndpoint` e a configuração de `TargetNeptuneClusterPort` será ignorada.
- **`BlockSparqlReplicationOnBlankNode`** — Sinalizador booleano que, se definido como`true`, interrompe a replicação de dados BlankNode em SPARQL (RDF). O valor padrão é `false`.

Alarme

- **`Required to create Cloud watch Alarm`**— Defina isso `true` se quiser criar um CloudWatch alarme para a nova pilha.
- **`SNS Topic ARN for Cloudwatch Alarm Notifications`**— O ARN do tópico do SNS para o CloudWatch qual as notificações de alarme devem ser enviadas (necessário somente se os alarmes estiverem habilitados).
- **`Email for Alarm Notifications`**: o endereço de e-mail para o qual as notificações de alarme devem ser enviadas (somente necessário se os alarmes estiverem habilitados).

Para o destino da notificação de alarmes, é possível adicionar somente SNS, somente e-mail ou SNS e e-mail.

Execute o AWS CloudFormation modelo

Agora é possível concluir o processo de provisionamento de uma instância de aplicação do consumidor de fluxos do Neptune da seguinte forma:

1. Em AWS CloudFormation, na página Especificar detalhes da pilha, escolha Avançar.
2. Na página Options (Opções), escolha Next (Avançar).

3. Na página Revisar, marque a primeira caixa de seleção para confirmar que o AWS CloudFormation criará recursos do IAM. Marque a segunda caixa de seleção para confirmar `CAPABILITY_AUTO_EXPAND` para a nova pilha.

   > **Note**
   >
   > `CAPABILITY_AUTO_EXPAND` confirma explicitamente que os macros serão expandidos ao criar a pilha, sem revisão anterior. Os usuários geralmente criam um conjunto de alterações a partir de um modelo processado para que as alterações feitas pelos macros possam ser revisadas antes de criar a pilha. Para obter mais informações, consulte a AWS CloudFormation CreateStackAPI na Referência AWS CloudFormation da API.

   Em seguida, selecione Criar.

Como atualizar o instrumento de sondagem de fluxos com os artefatos mais recentes do Lambda

É possível atualizar o instrumento de sondagem de fluxos com os artefatos de código mais recentes do Lambda:

1. No AWS Management Console, navegue até a AWS CloudFormation pilha principal AWS CloudFormation e selecione-a.
2. Selecione a opção Atualizar para a pilha.
3. Selecione Substituir modelo atual.
4. Para a origem do modelo, escolha o URL do Amazon S3 e insira o seguinte URL do S3:

   ```
   https://aws-neptune-customer-samples.s3.amazonaws.com/neptune-stream/
   neptune_to_neptune.json
   ```

5. Selecione Avançar sem alterar nenhum AWS CloudFormation parâmetro.
6. Escolha Update Stack (Atualizar pilha).

A pilha agora atualizará os artefatos do Lambda com os mais recentes.

Usar a replicação entre regiões de fluxos do Neptune para recuperação de desastres

O Neptune oferece duas maneiras de implementar recursos de failover entre regiões:

- Cópia e restauração de snapshots entre regiões
- Usar fluxos do Neptune para replicar dados entre dois clusters em duas regiões diferentes.

A cópia e a restauração de snapshots entre regiões têm a menor sobrecarga operacional para recuperar um cluster do Neptune em uma região diferente. No entanto, copiar um snapshot entre regiões pode exigir um tempo significativo de transferência de dados, já que um snapshot é um backup completo do cluster do Neptune. Como resultado, a cópia e a restauração de snapshots entre regiões podem ser usadas para situações que exigem apenas um objetivo de ponto de recuperação (RPO) de horas e um objetivo de tempo de recuperação (RTO) de horas.

Um objetivo de ponto de recuperação (RPO) é medido pelo tempo entre os backups. Ele define quantos dados podem ser perdidos entre o momento em que o último backup foi feito e o momento em que o banco de dados foi recuperado.

Um objetivo de tempo de recuperação (RTO) é medido pelo tempo necessário para realizar uma operação de recuperação. Esse é o tempo necessário para que o cluster de banco de dados realize o failover em um banco de dados recuperado após a ocorrência de uma falha.

Os fluxos do Neptune oferecem uma maneira de manter um cluster de backup do Neptune sempre sincronizado com o cluster de produção principal. Se ocorrer uma falha, o banco de dados fará o failover para o cluster de backup. Isso reduz o RPO e o RTO para minutos, já que os dados são constantemente copiados no cluster de backup, que está imediatamente disponível como destino de failover a qualquer momento.

A desvantagem de usar os fluxos do Neptune dessa forma é que tanto a sobrecarga operacional necessária para manter os componentes de replicação quanto o custo de ter um segundo cluster de banco de dados do Neptune on-line o tempo todo podem ser significativos.

Configurar a replicação de Neptune-para-Neptune

O cluster de banco de dados de produção principal reside em uma VPC em uma região de origem específica. Há três itens principais que você precisa replicar ou emular em uma região de recuperação diferente para fins de recuperação de desastres:

- Os dados armazenados no cluster.
- A configuração do cluster principal. Isso incluiria se ele usa a autenticação do IAM, se é criptografado, os parâmetros de cluster de banco de dados, os parâmetros de instância, os tamanhos de instância, etc.).
- A topologia de rede que ele usa, incluindo a VPC de destino, os grupos de segurança, etc.

Você pode usar as APIs de gerenciamento do Neptune, como as seguintes, para coletar essas informações:

- `DescribeDBClusters`
- `DescribeDBInstances`
- `DescribeDBClusterParameters`
- `DescribeDBParameters`
- `DescribeVpcs`

Com as informações coletadas, é possível usar o procedimento a seguir para configurar um cluster de backup em uma região diferente, para a qual o cluster de produção pode fazer failover em caso de falha.

1: Habilitar fluxos do Neptune

É possível usar o ModifyDBClusterParameterGroup para definir o parâmetro `neptune_streams` como 1. Assim, reinicialize todas as instâncias no cluster de banco de dados para que a alteração tenha efeito.

É uma boa ideia realizar pelo menos uma operação de adição ou atualização no cluster de banco de dados de origem após a ativação dos fluxos do Neptune. Isso preenche o fluxo de alterações com pontos de dados que podem ser referenciados posteriormente ao ressincronizar o cluster de produção com o cluster de backup.

2: Criar uma VPC na região em que você deseja configurar o cluster de backup

Antes de criar um cluster de banco de dados do Neptune em uma região diferente do cluster principal, você precisa estabelecer uma nova VPC na região de destino para hospedar o cluster. A conectividade entre os clusters principal e de backup é estabelecida por meio do emparelhamento da VPC, que usa tráfego em sub-redes privadas em diferentes VPCs. No entanto, para estabelecer o emparelhamento da VPC entre duas VPCs, elas não devem ter blocos CIDR nem espaços de

endereço IP sobrepostos. Isso significa que você não pode simplesmente usar a VPC padrão nas duas regiões, porque o bloco CIDR para uma VPC padrão é sempre o mesmo (`172.31.0.0/16`).

Você pode usar uma VPC existente na região de destino, desde que ela atenda às seguintes condições:

- Ela não tem um bloco CIDR que se sobreponha ao bloco CIDR da VPC na qual o cluster principal está localizado.
- Ela ainda não está emparelhado com outra VPC que tenha o mesmo bloco CIDR da VPC em que seu cluster principal está localizado.

Se não houver uma VPC adequada disponível na região de destino, crie uma usando a API `CreateVpc` do Amazon EC2.

3: Criar um snapshot do cluster principal e restaure-o na região de backup de destino

Agora você cria um cluster do Neptune em uma VPC apropriada na região de backup de destino que é uma cópia do cluster de produção:

Fazer uma cópia do cluster de produção na região de backup

1. Na sua região de backup de destino, recrie os parâmetros e grupos de parâmetros usados pelo cluster de banco de dados de produção. Você pode fazer isso usando `CreateDBClusterParameterGroup`, `CreateDBParameterGroup`, `ModifyDBClusterParameterGroup` e `ModifyDBParameterGroup`.

   Observe que as APIs `CopyDBClusterParameterGroup` e `CopyDBParameterGroup` e atualmente não são compatíveis com a cópia entre regiões.
2. Use `CreateDBClusterSnapshot` para criar um snapshot do cluster de produção na VPC na região de produção.
3. Use `CopyDBClusterSnapshot` para copiar o snapshot na VPC na região de backup de destino.
4. Use `RestoreDBClusterFromSnapshot` para criar um cluster de banco de dados na VPC na região de backup de destino usando o snapshot copiado. Use as configurações e os parâmetros que você copiou do cluster de produção principal.
5. O novo cluster do Neptune agora existe, mas não contém nenhuma instância. Use `CreateDBInstance` para criar uma instância principal/de gravador que tenha o mesmo tipo e

tamanho da instância de gravador do cluster de produção. No momento, não há necessidade de criar réplicas de leitura adicionais, a menos que a instância de backup seja usada para atender à E/S de leitura na região de destino antes de um failover.

4: Estabelecer o emparelhamento da VPC entre a VPC do cluster principal e a VPC do novo cluster de backup

Ao configurar o emparelhamento da VPC, você permite que a VPC do cluster principal se comunique com a VPC do cluster de backup como se fosse uma única rede privada. Para fazer isso, siga as seguintes etapas:

1. Na VPC do cluster de produção, chame a API `CreateVpcPeeringConnection` para estabelecer a conexão de emparelhamento.
2. Na VPC do cluster de backup de destino, chame a API `AcceptVpcPeeringConnection` para aceitar a conexão de emparelhamento.
3. Na VPC do cluster de produção, use a API `CreateRoute` para adicionar uma rota à tabela de rotas da VPC que redireciona todo o tráfego para o bloco CIDR da VPC de destino para que ela use a lista de prefixos de emparelhamento da VPC.
4. Da mesma forma, a partir da VPC do cluster de backup de destino, use a API `CreateRoute` para adicionar uma rota à tabela de rotas da VPC que direciona o tráfego para a VPC do cluster principal.

5: Configurar a infraestrutura de replicação de fluxos do Neptune

Agora que os dois clusters foram implantados e a comunicação de rede entre as duas regiões foi estabelecida, use o AWS CloudFormation modelo Neptune-to-Neptune para implantar a função Lambda do consumidor do Neptune Streams com a infraestrutura adicional que suporta a replicação de dados. Faça isso na VPC do cluster de produção principal.

Os parâmetros que você precisará fornecer para essa AWS CloudFormation pilha são:

- **`NeptuneStreamEndpoint`**: o endpoint de fluxos para o cluster principal, em formato de URL. Por exemplo: `https://(cluster name):8182/pg/stream`.
- **`QueryEngine`**: deve ser `gremlin`, `sparql` ou `openCypher`.
- **`RouteTableIds`**: permite adicionar rotas para um endpoint da VPC do DynamoDB e um endpoint da VPC de monitoramento.

Dois parâmetros adicionais, ou seja, `CreateMonitoringEndpoint` e `CreateDynamoDBEndpoint`, também deverão ser definidos como verdadeiros se ainda não existirem na VPC do cluster principal. Se eles já existirem, verifique se estão definidos como falsos ou a AWS CloudFormation criação falhará.

- **`SecurityGroupIds`**: especifica o grupo de segurança usado pelo consumidor do Lambda para se comunicar com o endpoint de fluxos do Neptune do cluster principal.

  No cluster de backup de destino, anexe um grupo de segurança que viabilize o tráfego proveniente desse grupo de segurança.
- **`SubnetIds`**: uma lista de IDs de sub-rede na VPC do cluster principal que pode ser usada pelo consumidor do Lambda para se comunicar com o cluster principal.
- **`TargetNeptuneClusterEndpoint`**: o endpoint do cluster (somente nome do host) do cluster de backup de destino.
- **`TargetAWSRegion`**— A AWS região do cluster de backup de destino, como `us-east-1`). Você deve fornecer esse parâmetro somente quando a AWS região do cluster de backup de destino for diferente da região do cluster de origem do Neptune, como no caso da replicação entre regiões. Se as regiões de origem e de destino forem iguais, esse parâmetro será opcional.

  Observe que, se o `TargetAWSRegion` valor não for uma AWS região válida compatível com o Neptune, o processo falhará.
- **`VPC`**: o ID da VPC do cluster principal.

Todos os outros parâmetros podem ser deixados com seus valores padrão.

Depois que o AWS CloudFormation modelo for implantado, o Neptune começará a replicar todas as alterações do cluster primário para o cluster de backup. Você pode monitorar essa replicação nos CloudWatch registros gerados pela função de consumidor do Lambda.

Outras considerações

- Se precisar usar a autenticação do IAM entre os clusters primário e de backup, você também pode configurá-la ao invocar o AWS CloudFormation modelo.
- Se a criptografia em repouso estiver habilitada no cluster principal, pense em como gerenciar as chaves do KMS associadas ao copiar o snapshot na região de destino e associar uma nova chave do KMS na região de destino.

- Uma prática recomendada é usar CNAMEs de DNS na frente dos endpoints do Neptune usados em suas aplicações. Então, se você precisar fazer o failover manual para o cluster de backup de destino, esses CNAMEs poderão ser alterados para apontar para o cluster de destino e/ou endpoints da instância.

Pesquisa de texto completo no Amazon Neptune usando o Amazon Service OpenSearch

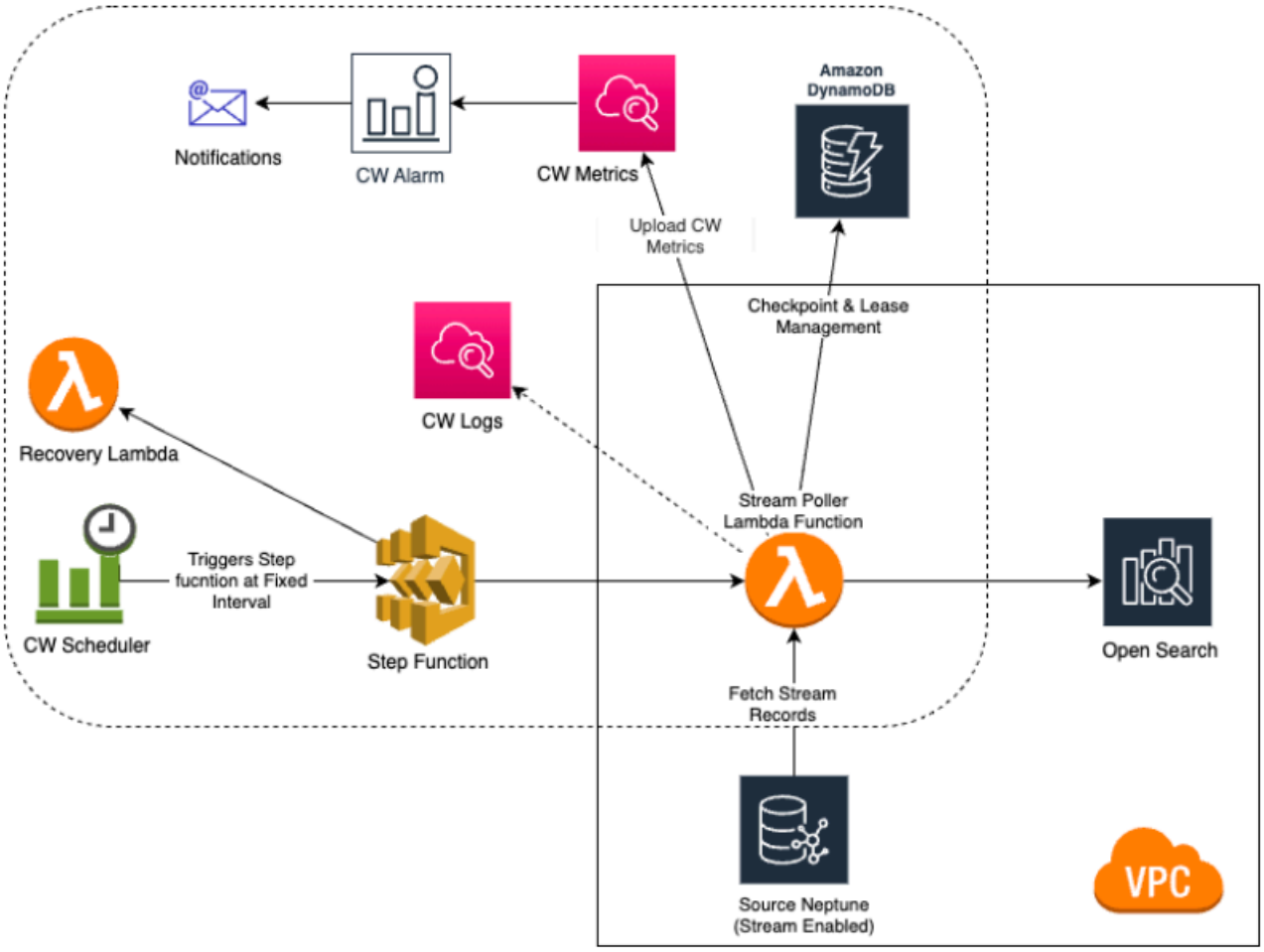
O Neptune se integra ao OpenSearch Amazon Service OpenSearch (Service) para oferecer suporte à pesquisa de texto completo nas consultas Gremlin e SPARQL. Esse atributo está disponível a partir da versão 1.0.2.1 do mecanismo do Neptune, embora seja recomendável usá-lo com a versão do mecanismo `1.0.4.2` ou posterior para aproveitar as correções mais recentes.

A partir da versão 1.3.0.0 do mecanismo, o Amazon Neptune oferece suporte ao uso do OpenSearch Amazon Service Serverless para pesquisa de texto completo em consultas Gremlin e SPARQL.

> **Note**
>
> Ao se integrar com o Amazon OpenSearch Service, o Neptune exige a versão 7.1 ou superior do Elasticsearch e funciona com 2.3, 2.5 e superior. OpenSearch O Neptune também funciona com o Serverless. OpenSearch

Você pode usar o Neptune com um cluster de serviços OpenSearch existente que tenha sido preenchido de acordo com o. Modelo de dados do Neptune para dados do OpenSearch Ou você pode criar um domínio de OpenSearch serviço vinculado ao Neptune usando uma pilha. AWS CloudFormation

> **Important**
>
> O processo de replicação de Netuno OpenSearch para replicação descrito aqui não replica nós em branco. Essa é uma limitação importante a ser observada.
> Além disso, se você habilitar o controle de acesso refinado em seu OpenSearch cluster, também precisará habilitar a autenticação do IAM em seu banco de dados Neptune.



Tópicos

- Replicação do Amazon Neptune para OpenSearch
- Replicação para o OpenSearch Sem Servidor
- Consultar a partir de um cluster do OpenSearch com controle de acesso refinado (FGAC) habilitado
- Usar a sintaxe de consulta Apache Lucene em consultas de pesquisa de texto completo do Neptune
- Modelo de dados do Neptune para dados do OpenSearch
- Parâmetros de pesquisa de texto completo do Neptune
- Indexação sem string do OpenSearch no Amazon Neptune
- Execução de consulta de pesquisa de texto completo no Amazon Neptune
- Exemplos de consulta do SPARQL usando a pesquisa de texto completo no Neptune
- Usar a pesquisa de texto completo do Neptune em consultas do Gremlin

- Solução de problemas da pesquisa de texto completo do Neptune

Replicação do Amazon Neptune para OpenSearch

O Amazon Neptune oferece suporte à pesquisa de texto completo em consultas Gremlin e SPARQL usando o Amazon Service (Service). OpenSearch OpenSearch Você pode usar uma AWS CloudFormation pilha para vincular um domínio de OpenSearch serviço ao Neptune. O AWS CloudFormation modelo cria uma instância de aplicativo para consumidores de fluxos que fornece replicação do Neptune para-. OpenSearch

Antes de começar, você precisa de um cluster de banco de dados Neptune existente com fluxos habilitados para servir como origem e de OpenSearch um domínio de serviço para servir como destino de replicação.

Se você já tem um domínio de OpenSearch serviço de destino existente que pode ser acessado pelo Lambda na VPC em que seu cluster de banco de dados Neptune está localizado, o modelo pode usar esse. Caso contrário, será necessário criar um.

> **Note**
>
> O OpenSearch cluster e a função Lambda que você cria devem estar localizados na mesma VPC do seu cluster de banco de dados Neptune, e o cluster OpenSearch deve estar configurado no modo VPC (não no modo Internet).

Recomendamos que você use uma instância recém-criada do Neptune para usar com o Service. OpenSearch Se você usa uma instância existente que já contém dados, você deve realizar uma sincronização de dados de OpenSearch serviço antes de fazer consultas, caso contrário, pode haver inconsistências de dados. Este GitHub projeto fornece um exemplo de como realizar a sincronização: Exportar Neptune para (https://github.com/awslabs/ /tree/master/ OpenSearch). amazon-neptune-tools export-neptune-to-elasticsearch

> **Important**
>
> Ao se integrar com o Amazon OpenSearch Service, o Neptune exige a versão 7.1 ou superior do Elasticsearch e funciona com as versões OpenSearch 2.3, 2.5 e futuras compatíveis do Opensearch.

> **Note**
>
> A partir da versão 1.3.0.0 do mecanismo, o Amazon Neptune oferece suporte ao uso do OpenSearch Amazon Service Serverless para pesquisa de texto completo em consultas Gremlin e SPARQL.

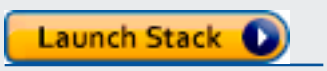
Tópicos

Usando um AWS CloudFormation modelo para iniciar a replicação do Neptune-to-- OpenSearch

Lance uma AWS CloudFormation pilha específica para sua região

Cada um dos AWS CloudFormation modelos abaixo cria uma instância de aplicativo streams-consumer em uma região específica AWS . Para iniciar a pilha correspondente usando o AWS CloudFormation console, escolha um dos botões Iniciar pilha na tabela a seguir, dependendo da AWS região que você deseja usar.

Região	Visualização	Visualizar no Designer	Executar
Leste dos EUA (Norte da Virgínia)	Visão	Visualizar no Designer	Launch Stack
Leste dos EUA (Ohio)	Visão	Visualizar no Designer	Launch Stack
Oeste dos EUA (N. da Califórnia)	Visão	Visualizar no Designer	Launch Stack

Região	Visualização	Visualizar no Designer	Executar
Oeste dos EUA (Oregon)	Visão	Visualizar no Designer	
Canadá (Central)	Visão	Visualizar no Designer	
América do Sul (São Paulo)	Visão	Visualizar no Designer	
Europa (Estocolmo)	Visão	Visualizar no Designer	
Europa (Irlanda)	Visão	Visualizar no Designer	
Europa (Londres)	Visão	Visualizar no Designer	
Europa (Paris)	Visão	Visualizar no Designer	
Europa (Frankfurt)	Visão	Visualizar no Designer	
Oriente Médio (Barém)	Visão	Visualizar no Designer	
Oriente Médio (Emirados Árabes Unidos)	Visão	Visualizar no Designer	
Israel (Tel Aviv)	Visão	Visualizar no Designer	
África (Cidade do Cabo)	Visão	Visualizar no Designer	

Região	Visualização	Visualizar no Designer	Executar
Ásia-Pacífico (Hong Kong)	Visão	Visualizar no Designer	
Ásia-Pacífico (Tóquio)	Visão	Visualizar no Designer	
Ásia-Pacífico (Seul)	Visão	Visualizar no Designer	
Ásia-Pacífico (Singapura)	Visão	Visualizar no Designer	
Ásia-Pacífico (Mumbai)	Visão	Visualizar no Designer	
China (Pequim)	Visão	Visualizar no Designer	
China (Ningxia)	Visão	Visualizar no Designer	
AWS GovCloud (Oeste dos EUA)	Visão	Visualizar no Designer	
AWS GovCloud (Leste dos EUA)	Visão	Visualizar no Designer	

Na página Create Stack (Criar pilha), selecione Next (Avançar).

Adicione detalhes sobre a nova OpenSearch pilha que você está criando

A página Especificar detalhes da pilha fornece propriedades e parâmetros que você pode usar para controlar a configuração da pesquisa de texto completo.

Nome da pilha — O nome da nova AWS CloudFormation pilha que você está criando. Geralmente, você pode usar o valor padrão, `NeptuneStreamPoller`.

Em Parameters (Parâmetros), forneça o seguinte:

Configuração de rede para a VPC na qual o consumidor de fluxos é executado

- **`VPC`**: forneça o nome da VPC na qual a função do Lambda de sondagem será executada.
- **`List of Subnet IDs`**: as sub-redes para as quais será estabelecida uma interface de rede. Adicione sub-redes correspondentes ao cluster do Neptune.
- **`List of Security Group Ids`**: forneça os IDs dos grupos de segurança que concedem o acesso de entrada de gravação ao cluster de banco de dados de origem do Neptune.
- **`List of Route Table Ids`**: é necessário para criar um endpoint do Amazon DynamoDB na VPC do Neptune, caso ainda não tenha um. Forneça uma lista separada por vírgulas de IDs de tabela de rotas associados às sub-redes.
- **`Require to create Dynamo DB VPC Endpoint`**: um valor booliano predefinido como `true`. Você só precisará alterá-lo para `false` se já tiver criado um endpoint do DynamoDB na VPC.
- **`Require to create Monitoring VPC Endpoint`**: um valor booliano predefinido como `true`. Você só precisará alterá-lo para `false` se já tiver criado um endpoint de monitoramento na VPC.

Instrumento de sondagem de fluxos

- **`Application Name`**: geralmente, você pode manter a configuração padrão (`NeptuneStream`). Se você usar um nome diferente, ele deverá ser exclusivo.
- **`Memory size for Lambda Poller`**: usado para definir o tamanho da memória disponível para a função do instrumento de sondagem do Lambda. O valor padrão é de 2.048 megabytes.
- **`Lambda Runtime`**: a linguagem usada na função do Lambda que recupera os itens do fluxo do Neptune. É possível definir como `python3.9` ou `java8`.
- **`S3 Bucket having Lambda code artifacts`**: deixe em branco, a menos que esteja usando uma função de sondagem personalizada do Lambda, carregada em um bucket do S3 diferente.
- **`S3 Key corresponding to Lambda Code artifacts`**: deixe em branco, a menos que esteja usando uma função de sondagem personalizada do Lambda.
- **`StartingCheckpoint`**: o ponto de verificação inicial do instrumento de sondagem de fluxos. O padrão é `0:0`, o que significa começar do início do fluxo do Neptune.
- **`StreamPollerInitialState`**: o estado inicial do instrumento de sondagem. O padrão é `ENABLED`, o que significa que a replicação de fluxos começará assim que a criação de toda a pilha for concluída.
- **`Logging level for Lambda`**: em geral, mantenha o valor padrão, `INFO`.

- **`Managed Policies for Lambda Execution`**: em geral, deixe em branco, a menos que esteja usando uma função de sondagem personalizada do Lambda.
- **`Stream Records Handler`**: em geral, deixe em branco, a menos que esteja usando um manipulador personalizado para os registros nos fluxos do Neptune.
- **`Maximum records Fetched from Stream`**: é possível usar esse parâmetro para ajustar o desempenho. Recomendamos iniciar com o valor padrão (`100`). O máximo permitido é 10.000. Quanto maior o número, menor serão as chamadas de rede necessárias para ler os registros do fluxo, mas maior será a memória necessária para processar os registros.
- **`Max wait time between two Polls (in Seconds)`**: determina com que frequência o instrumento de sondagem do Lambda será invocado para sondar os fluxos do Neptune. Defina esse valor como 0 para a sondagem contínua. O valor máximo é de 3.600 segundos (1 hora). Recomendamos o valor padrão (60 segundos) para começar, dependendo da rapidez com que os dados do gráfico mudam.
- **`Maximum Continuous polling period (in Seconds)`**: usado para definir um tempo limite para a função de sondagem do Neptune. Deve ser entre 5 segundos e 900 segundos. Recomendamos começar com o valor padrão (600 segundos).
- **`Step Function Fallback Period`**— O número de step-function-fallback-period unidades que aguardam o pesquisador, após o qual a função de etapa é chamada por meio do Amazon CloudWatch Events para se recuperar de uma falha. Recomendamos iniciar com o valor padrão (5 minutos).
- **`Step Function Fallback Period Unit`**: as unidades de tempo usadas para medir o `Step Function Fallback Period` anterior (minutos, horas, dias). Em geral, o padrão (minutos) é suficiente.
- **`Data replication scope`**— Determina se deve replicar os nós e as bordas, ou somente os nós para OpenSearch (isso se aplica somente aos dados do mecanismo Gremlin). Recomendamos começar com o valor padrão (All).
- **`Ignore OpenSearch missing document error`**— Sinalize para determinar se um erro de documento ausente OpenSearch pode ser ignorado. Erros de documentos ausentes ocorrem raramente, mas precisarão de intervenção manual se não forem ignorados. Em geral, o valor padrão (`True`) é um bom ponto de partida.
- **`Enable Non-String Indexing`**: sinalizador para habilitar ou desabilitar a indexação de campos que não têm conteúdo de string. Se esse sinalizador estiver definido como`true`, campos que não sejam de string serão indexados ou OpenSearch, se`false`, somente campos de string serão indexados. O padrão é `true`.

- **`Properties to exclude from being inserted into OpenSearch`**— Uma lista delimitada por vírgulas de chaves de propriedade ou predicado a serem excluídas da indexação. OpenSearch Se esse valor do parâmetro do CFN for deixado em branco, todas as chaves de propriedade serão indexadas.
- **`Datatypes to exclude from being inserted into OpenSearch`**— Uma lista delimitada por vírgula de tipos de dados de propriedades ou predicados a serem excluídos da indexação. OpenSearch Se esse valor do parâmetro CFN for deixado em branco, todos os valores de propriedade que podem ser convertidos com segurança em tipos de OpenSearch dados serão indexados.

Fluxo do Neptune

- **`Endpoint of source Neptune Stream`**: (obrigatório) assume um dos dois formatos:
  - **`https://`*`your DB cluster`*`:`*`port`*`/propertygraph/stream`** (ou o alias, `https://`*`your DB cluster`*`:`*`port`*`/pg/stream`).
  - `https://`*`your DB cluster`*`:`*`port`*`/sparql/stream`
- **`Neptune Query Engine`**: selecione Gremlin ou SPARQL.
- **`Is IAM Auth Enabled?`**: se o cluster de banco de dados do Neptune estiver usando a autenticação do IAM, defina esse parâmetro como `true`.
- **`Neptune Cluster Resource Id`**: se o cluster de banco de dados do Neptune estiver usando a autenticação do IAM, defina esse parâmetro como o ID do recurso do cluster. O ID do recurso não é igual ao ID do cluster. Em vez disso, o formato é: `cluster-` seguido por 28 caracteres alfanuméricos. Ele pode ser encontrado em Detalhes do cluster no console do Neptune.

OpenSearch Cluster de destino

- **`Endpoint for OpenSearch service`**— (Obrigatório) Forneça o endpoint para o OpenSearch serviço em sua VPC.
- **`Number of Shards for OpenSearch Index`**: o valor padrão (cinco) geralmente é um bom ponto de partida.
- **`Number of Replicas for OpenSearch Index`**: o valor padrão (um) geralmente é um bom ponto de partida.
- **`Geo Location Fields for Mapping`**: se você estiver usando os campos de geolocalização, liste as chaves de propriedade aqui.

Alarme

- **`Require to create Cloud watch Alarm`**— Defina isso `true` se quiser criar um CloudWatch alarme para a nova pilha.
- **`SNS Topic ARN for Cloudwatch Alarm Notifications`**— O ARN do tópico do SNS para o CloudWatch qual as notificações de alarme devem ser enviadas (necessário somente se os alarmes estiverem habilitados).
- **`Email for Alarm Notifications`**: o endereço de e-mail para o qual as notificações de alarme devem ser enviadas (somente necessário se os alarmes estiverem habilitados).

Para o destino da notificação de alarmes, é possível adicionar somente SNS, somente e-mail ou SNS e e-mail.

Execute o AWS CloudFormation modelo

Agora é possível concluir o processo de provisionamento de uma instância de aplicação do consumidor de fluxos do Neptune da seguinte forma:

1. Em AWS CloudFormation, na página Especificar detalhes da pilha, escolha Avançar.
2. Na página Options (Opções), escolha Next (Avançar).
3. Na página Revisar, marque a primeira caixa de seleção para confirmar que o AWS CloudFormation criará recursos do IAM. Marque a segunda caixa de seleção para confirmar `CAPABILITY_AUTO_EXPAND` para a nova pilha.

   > **Note**
   >
   > `CAPABILITY_AUTO_EXPAND` confirma explicitamente que os macros serão expandidos ao criar a pilha, sem revisão anterior. Os usuários geralmente criam um conjunto de alterações a partir de um modelo processado para que as alterações feitas pelos macros possam ser revisadas antes de criar a pilha. Para obter mais informações, consulte a operação AWS CloudFormation CreateStackda API na Referência AWS CloudFormation da API.

   Em seguida, selecione Criar.

Habilitar a pesquisa de texto completo em bancos de dados Neptune existentes

Se for possível pausar as workloads de gravação

A melhor maneira de habilitar a pesquisa de texto completo em um banco de dados Neptune existente geralmente é a seguinte, desde que seja possível pausar as workloads de gravação. Isso requer a criação de um clone, a ativação dos fluxos usando um parâmetro de cluster e a reinicialização de todas as instâncias. Criar um clone é uma operação relativamente rápida, portanto, o tempo de inatividade necessário é limitado.

As etapas necessárias são:

1. Interromper todas as workloads de gravação no banco de dados.
2. Habilitar fluxos no banco de dados (consulte Enabling Neptune Streams).
3. Criar um clone do banco de dados (consulte Database Cloning in Neptune).
4. Retomar as workloads de gravação.
5. Use a export-neptune-to-elasticsearchferramenta no github para realizar uma sincronização única do banco de dados clonado para o domínio. OpenSearch
6. Use o modelo do AWS CloudFormation da região para iniciar a sincronização do banco de dados original com atualização contínua (nenhuma alteração na configuração é necessária no modelo).
7. Exclua o banco de dados clonado e a AWS CloudFormation pilha criada para a `export-neptune-to-elasticsearch` ferramenta.

Se não for possível pausar as workloads de gravação

Se você não puder suspender as workloads de gravação no banco de dados, veja uma abordagem que exige ainda menos tempo de inatividade do que a abordagem recomendada acima, mas precisa ser realizada com cuidado:

1. Habilitar fluxos no banco de dados (consulte Enabling Neptune Streams).
2. Criar um clone do banco de dados (consulte Database Cloning in Neptune).
3. Obtenha o `eventID` mais recente dos fluxos no banco de dados clonado executando um comando desse tipo no endpoint da API de fluxos (consulte Calling the Neptune Streams REST API para obter mais informações):

```
curl "https://(your neptune endpoint):(port)/(propertygraph or sparql)/stream?
iteratorType=LATEST"
```

Anote os valores nos campos `commitNum` e `opNum` no objeto `lastEventId` na resposta.

4. Use a export-neptune-to-elasticsearchferramenta no github para realizar uma sincronização única do banco de dados clonado para o domínio. OpenSearch
5. Use o modelo do AWS CloudFormation da região para iniciar a sincronização do banco de dados original com atualização contínua.

   Faça a seguinte alteração ao criar a pilha: na página de detalhes da pilha, na seção Parâmetros, defina o valor do campo `StartingCheckpoint` como *`commitNum`*:*`opnum`* usando os valores `commitNum` e `opNum` que você registrou acima.
6. Exclua o banco de dados clonado e a AWS CloudFormation pilha criada para a `export-neptune-to-elasticsearch` ferramenta.

Atualizar o instrumento de sondagem de fluxos

Como atualizar o instrumento de sondagem de fluxos com os artefatos mais recentes do Lambda

É possível atualizar o instrumento de sondagem de fluxos com os artefatos de código mais recentes do Lambda:

1. No AWS Management Console, navegue até a AWS CloudFormation pilha principal AWS CloudFormation e selecione-a.
2. Selecione a opção Atualizar para a pilha.
3. Selecione Substituir modelo atual.
4. Para a origem do modelo, escolha o URL do Amazon S3 e insira o seguinte URL do S3:

   ```
   https://aws-neptune-customer-samples.s3.amazonaws.com/neptune-stream/
   neptune_to_elastic_search.json
   ```
5. Selecione Avançar sem alterar nenhum AWS CloudFormation parâmetro.
6. Escolha Update Stack (Atualizar pilha).

A pilha agora atualizará os artefatos do Lambda com os mais recentes.

Estender o instrumento de sondagem de fluxos para oferecer compatibilidade com campos personalizados

O instrumento de sondagem de fluxos atual pode ser facilmente estendido para escrever código personalizado para lidar com campos personalizados, conforme explicado em detalhes nesta postagem no blog: Capture graph changes using Neptune Streams.

> **Note**
>
> Ao adicionar um campo personalizado OpenSearch, certifique-se de adicionar o novo campo como um objeto interno de um predicado (consulteModelo de dados de pesquisa de texto completo do Neptune).

Desabilitar e reabilitar o processo do instrumento de sondagem de fluxos

> **Warning**
>
> Tenha cuidado ao desabilitar o processo do instrumento de sondagem de fluxos. Poderá ocorrer perda de dados se o processo for pausado por mais tempo do que a janela de validade de fluxos. A janela padrão é de sete dias, mas a partir da versão 1.2.0.0 do mecanismo, é possível definir uma janela de validade de fluxos personalizada de até noventa dias.

Desabilitar (pausar) o processo do instrumento de sondagem de fluxos

1. Faça login no AWS Management Console e abra o EventBridge console da Amazon em https://console.aws.amazon.com/events/.
2. Selecione Regras no painel de navegação.
3. Selecione a regra cujo nome contém o nome que você forneceu como Nome do aplicativo no AWS CloudFormation modelo que você usou para configurar o stream poller.
4. Escolha Disable.
5. Abra o console do Step Functions em https://console.aws.amazon.com/states/.

6. Selecione a função da etapa de execução correspondente ao processo do instrumento de sondagem de fluxos. Novamente, o nome dessa função de etapa contém o nome que você forneceu como Nome do aplicativo no AWS CloudFormation modelo usado para configurar o stream poller. É possível filtrar por status de execução da função para ver somente as funções em execução.
7. Escolha Parar.

Reabilitar o processo do instrumento de sondagem de fluxos

1. Faça login no AWS Management Console e abra o EventBridge console da Amazon em https://console.aws.amazon.com/events/.
2. Selecione Regras no painel de navegação.
3. Selecione a regra cujo nome contém o nome que você forneceu como Nome do aplicativo no AWS CloudFormation modelo que você usou para configurar o stream poller.
4. Escolha Disable. A regra do evento com base no intervalo programado especificado agora acionará uma nova execução da função de etapa.

Replicação para o OpenSearch Sem Servidor

A partir da versão 1.3.0.0 do mecanismo, o Amazon Neptune oferece suporte ao uso do Amazon OpenSearch Service Sem Servidor para pesquisa de texto completo em consultas Gremlin e SPARQL.

Se você estiver replicando para o OpenSearch Sem Servidor, adicione a função de execução do instrumento de sondagem de fluxo do Lambda à política de acesso a dados da coleção do OpenSearch Sem Servidor. O ARN da função de execução do instrumento de sondagem de fluxo do Lambda tem este formato:

```
arn:aws:iam::(account ID):role/stack-name-NeptuneOSReplication-
NeptuneStreamPollerExecu-(uuid)
```

Para obter mais informações, consulte Controle de acesso a dados do Amazon OpenSearch Sem Servidor.

Se você habilitou o controle de acesso refinado no cluster do OpenSearch, também precisará habilitar a autenticação do IAM no banco de dados Neptune.

A entidade do IAM (usuário ou perfil) usada para se conectar ao banco de dados Neptune deve ter permissões tanto para o Neptune quanto para a coleção do OpenSearch Sem Servidor. Isso significa que o usuário ou o perfil deve ter uma política do OpenSearch Sem Servidor como esta associada:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::(account ID):root"
      },
      "Action": "aoss:APIAccessAll",
      "Resource": "arn:aws:aoss:(region):(account ID):collection/(collection ID)"
    }
  ]
}
```

Consulte Declarações personalizadas de política de acesso aos dados do IAM para o Amazon Neptune para obter mais informações.

Consultar a partir de um cluster do OpenSearch com controle de acesso refinado (FGAC) habilitado

Se você habilitou o controle de acesso refinado no cluster do OpenSearch, também precisará habilitar a autenticação do IAM no banco de dados Neptune.

A entidade do IAM (usuário ou perfil) usada para se conectar ao banco de dados Neptune deve ter permissões tanto para o Neptune quanto para o cluster do OpenSearch. Isso significa que o usuário ou o perfil deve ter uma política do OpenSearch Service como esta associada:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::account-id:root"
      },
      "Action": "es:*",
```

```
      "Resource": "arn:aws:es:region:account-id:es-resource-id/*"
    }
  ]
}
```

Consulte Declarações personalizadas de política de acesso aos dados do IAM para o Amazon Neptune para obter mais informações.

Usar a sintaxe de consulta Apache Lucene em consultas de pesquisa de texto completo do Neptune

O OpenSearch é compatível com o uso da sintaxe Apache Lucene para consultas query_string. Isso é especialmente útil para transmitir vários filtros em uma consulta.

O Neptune usa uma estrutura aninhada para armazenar propriedades em um documento do OpenSearch (consulte Modelo de dados de pesquisa de texto completo do Neptune). Ao usar a sintaxe Lucene, é necessário usar caminhos completos para as propriedades no modelo aninhado.

Veja aqui um exemplo do Gremlin:

```
g.withSideEffect("Neptune#fts.endpoint", "es_endpoint")
 .withSideEffect("Neptune#fts.queryType", "query_string")
 .V()
 .has("*", "Neptune#fts predicates.name.value:\"Jane Austin\" AND entity_type:Book")
```

Veja um exemplo do SPARQL:

```
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
  neptune-fts:config neptune-fts:endpoint 'http://localhost:9200 (http://
localhost:9200/)' .
  neptune-fts:config neptune-fts:queryType 'query_string' .
  neptune-fts:config neptune-fts:query "predicates.\\*foaf\\*name.value:Ronak AND
 predicates.\\*foaf\\*surname.value:Sh*" .
  neptune-fts:config neptune-fts:field '*' .
  neptune-fts:config neptune-fts:return ?res .
}
```

Modelo de dados do Neptune para dados do OpenSearch

O Amazon Neptune usa uma estrutura de documento unificada em JSON para armazenar dados do SPARQL e do Gremlin no OpenSearch Service. Cada documento no OpenSearch corresponde a uma entidade e armazena todas as informações relevantes para essa entidade. Para o Gremlin, os vértices e as bordas são considerados entidades, portanto, os documentos correspondentes do OpenSearch têm informações sobre vértices, rótulos e propriedades. Para o SPARQL, os assuntos podem ser considerados entidades, portanto, os documentos do OpenSearch correspondentes têm informações sobre todos os pares de objeto-predicado em um documento.

> **Note**
>
> A implementação da replicação do Neptune para o OpenSearch armazena somente dados de string. No entanto, você pode modificá-la para armazenar outros tipos de dados.

A estrutura de documento unificada em JSON se parece com o que segue.

```
{
  "entity_id": "Vertex Id/Edge Id/Subject URI",
  "entity_type": [List of Labels/rdf:type object value],
  "document_type": "vertex/edge/rdf-resource"
  "predicates": {
    "Property name or predicate URI": [
      {
        "value": "Property Value or Object Value",
        "graph": "(Only for Sparql) Named Graph Quad is present"
        "language": "(Only for Sparql) rdf:langString"
      },
      {
        "value": "Property Value 2/ Object Value 2",
      }
    ]
  }
}
```

- `entity_id`: ID de entidade exclusivo que representa o documento.
  - Para o SPARQL, este é o URI do sujeito.
  - Para o Gremlin, este é o `Vertex_ID` ou `Edge_ID`.

- `entity_type`: representa um ou mais rótulos para um vértice ou uma borda, ou zero ou mais valores de predicado `rdf:type` para um assunto.
- `document_type`: usado para especificar se o documento atual representa um vértice, uma borda ou um recurso RDF.
- `predicates`: para o Gremlin, armazena propriedades e valores para um vértice ou uma borda. Para o SPARQL, ele armazena pares de predicado-objeto.

  O nome da propriedade assume o formato `properties.name.value` no OpenSearch. Para consultá-lo, é necessário nomeá-lo dessa forma.
- `value` : um valor de propriedade para o Gremlin ou um valor de objeto para o SPARQL.
- `graph`: um grafo nomeado para o SPARQL.
- `language`: uma tag de linguagem para um literal `rdf:langString` no SPARQL.

Exemplo de documento OpenSearch em SPARQL

Dados

```
@prefix dt:   <http://example.org/datatype#> .
@prefix ex:   <http://example.org/> .
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

ex:simone   rdf:type     ex:Person                   ex:g1
ex:michael  rdf:type     ex:Person                   ex:g1
ex:simone   ex:likes     "spaghetti"                 ex:g1

ex:simone   ex:knows     ex:michael                  ex:g2   # Not stored in ES
ex:simone   ex:likes     "spaghetti"                 ex:g2
ex:simone   ex:status    "La vita è un sogno"@it     ex:g2

ex:simone   ex:age       "40"^^xsd:int               DG      # Not stored in ES
ex:simone   ex:dummy     "testData"^^dt:newDataType  DG
ex:simone   ex:hates     _:bnode                             # Not stored in ES
_:bnode     ex:means     "coding"                    DG      # Not stored in ES
```

Documentos

```
{
  "entity_id": "http://example.org/simone",
```

```
  "entity_type": ["http://example.org/Person"],
  "document_type": "rdf-resource"
  "predicates": {
    "http://example.org/likes": [
      {
        "value": "spaghetti",
        "graph": "http://example.org/g1"
      },
      {
        "value": "spaghetti",
        "graph": "http://example.org/g2"
      }
    ]
    "http://example.org/status": [
      {
        "value": "La vita è un sogno",
        "language": "it"       // Only present for rdf:langString
      }
    ]
  }
}
```

```
{
  "entity_id" : "http://example.org/michael",
  "entity_type" : ["http://example.org/Person"],
  "document_type": "rdf-resource"
}
```

Exemplo de documento do OpenSearch em Gremlin

Dados

```
# Vertex 1
simone   label    Person       <== Label
simone   likes    "spaghetti"  <== Property
simone   likes    "rice"       <== Property
simone   age      40           <== Property

# Vertex 2
michael  label    Person       <== Label

# Edge 1
simone  knows     michael      <== Edge
```

```
e1      updated  "2019-07-03"  <== Edge Property
e1      through  "company"     <== Edge Property
e1      since     10           <== Edge Property
```

Documentos

```
{
  "entity_id": "simone",
  "entity_type": ["Person"],
  "document_type": "vertex",
  "predicates": {
    "likes": [
      {
        "value": "spaghetti"
      },
      {
        "value": "rice"
      }
    ]
  }
}
```

```
{
  "entity_id" : "michael",
  "entity_type" : ["Person"],
  "document_type": "vertex"
}
```

```
{
  "entity_id": "e1",
  "entity_type": ["knows"],
  "document_type": "edge"
  "predicates": {
    "through": [
      {
        "value": "company"
      }
    ]
  }
}
```

Parâmetros de pesquisa de texto completo do Neptune

O Amazon Neptune usa os seguintes parâmetros para especificar as consultas de texto completo do OpenSearch no Gremlin e no SPARQL:

- **`queryType`**: (obrigatório) o tipo de consulta do OpenSearch. (Para obter uma lista dos tipos de consulta, consulte a documentação do OpenSearch). O Neptune é compatível com os seguintes tipos de consulta do OpenSearch:
  - simple_query_string: exibe documentos baseados em uma string de consulta fornecida, usando um analisador com uma sintaxe Lucene limitada, mas tolerante a falhas. Este é o tipo de consulta padrão.

    Esta consulta usa uma sintaxe simples para analisar e dividir a string de consulta fornecida em termos baseados em operadores especiais. A consulta analisa todos os termos de forma independente antes de retornar os documentos correspondentes.

    Embora a sintaxe seja mais limitada que a consulta `query_string`, a consulta `simple_query_string` não retorna erros para sintaxe inválida. Em vez disso, ela ignora as partes inválidas da string de consulta.
  - match: a consulta `match` é a consulta padrão para realizar uma pesquisa de texto completo, incluindo opções para correspondência difusa.
  - prefix: exibe documentos que contêm um prefixo específico em um campo fornecido.
  - fuzzy: exibe documentos que contêm termos semelhantes ao termo de pesquisa, conforme medido por uma distância de edição de Levenshtein.

    Uma distância de edição é o número de alterações de um caractere necessárias para transformar um termo em outro. Essas alterações podem incluir:
    - Mudar um caractere (caixa para faixa).
    - Remover um caractere (preto para reto).
    - Inserir um caractere (acender para ascender).
    - Transposição de dois caracteres adjacentes (bolsa para bolas).

    Para encontrar termos semelhantes, a consulta difusa cria um conjunto de todas as possíveis variações e expansões do termo de pesquisa dentro de uma distância de edição especificada e retorna correspondências exatas para cada uma dessas variantes.

- term: exibe documentos que contêm uma correspondência exata de um termo definido em um dos campos especificados.

  Você pode usar a consulta `term` para encontrar documentos baseados em um valor preciso, como preço, ID de produto ou nome de usuário.

  > ⚠ Warning
  >
  > Evite usar a consulta de termo para campos de texto. Por padrão, o OpenSearch altera os valores de campos de texto como parte da análise, o que pode dificultar a localização de correspondências exatas para valores de campos de texto.
  > Para pesquisar valores de campo de texto, use a consulta de correspondência.

- query_string: exibe documentos baseados em uma string de consulta fornecida, usando um analisador com uma sintaxe estrita (sintaxe Lucene).

  Esta consulta usa uma sintaxe para analisar e dividir a string de consulta fornecida com base em operadores, como E ou NÃO. A consulta analisa cada texto dividido de forma independente antes de retornar os documentos correspondentes.

  Você pode usar a consulta `query_string` para criar uma pesquisa complexa que inclui caracteres curinga, pesquisas em vários campos, entre outros. Embora seja versátil, a consulta é restrita e retorna um erro se a string da consulta incluir uma sintaxe inválida.

  > ⚠ Warning
  >
  > Como ela retorna um erro para uma sintaxe inválida, não recomendamos o uso da consulta `query_string` para caixas de pesquisa.
  > Se você não precisar oferecer suporte para a sintaxe de consulta, considere usar a consulta `match`. Se você precisar dos recursos de uma sintaxe de consulta, use a consulta `simple_query_string`, que é menos estrita.

- **`field`**: o campo no OpenSearch no qual executar a pesquisa. Isto pode ser omitido somente se o `queryType` permitir (assim como o `simple_query_string` e o `query_string`), nesse caso, a pesquisa é realizada em todos os campos. No Gremlin, está implícito.

  É possível especificar múltiplos campos se a consulta permitir, assim como `simple_query_string` e `query_string`.

- **`query`**: (obrigatório) a consulta a ser executada no OpenSearch. O conteúdo deste campo pode variar de acordo com o queryType. Diferentes queryTypes aceitam sintaxes diferentes, como, por exemplo, o `Regexp`. No Gremlin, o `query` está implícito.
- **`maxResults`**: o número máximo de resultados a serem exibidos. O padrão é a configuração `index.max_result_window` do OpenSearch, que é padronizada como dez mil. O parâmetro `maxResults` pode especificar qualquer número inferior a esse.

  > **Important**
  >
  > Se você definir `maxResults` como um valor maior que o valor `index.max_result_window` do OpenSearch e tentar recuperar mais de resultados `index.max_result_window`, o OpenSearch exibirá um erro `Result window is too large`. No entanto, o Neptune processa o erro corretamente sem propagá-lo. Lembre-se disso se estiver tentando buscar mais do que `index.max_result_window` resultados.

- **`minScore`**: a pontuação mínima que um resultado de pesquisa pode ter para ser exibido. Consulte a documentação de relevância do OpenSearch para obter uma explicação da pontuação do resultado.
- **`batchSize`**: o Neptune sempre obtém dados em lotes (o tamanho de lote padrão é cem). É possível usar esse parâmetro para ajustar o desempenho. O tamanho do lote não pode exceder a configuração `index.max_result_window` do OpenSearch, que é predefinida como dez mil.
- **`sortBy`**: um parâmetro opcional que permite classificar os resultados exibidos pelo OpenSearch por um dos seguintes itens:
  - Um campo de string específico no documento –

    Por exemplo, em uma consulta SPARQL, é possível especificar:

    ```
    neptune-fts:config neptune-fts:sortBy foaf:name .
    ```

    Em uma consulta Gremlin, é possível especificar:

    ```
    .withSideEffect('Neptune#fts.sortBy', 'name')
    ```

  - Um campo que não seja de string específico (*`long`*, *`double`* etc.) no documento –

    Observe que, ao classificar em um campo que não seja de string, você precisa acrescentar `.value` ao nome do campo para diferenciá-lo de um campo de string.

Por exemplo, em uma consulta SPARQL, é possível especificar:

```
    neptune-fts:config neptune-fts:sortBy foaf:name.value .
```

Em uma consulta Gremlin, é possível especificar:

```
    .withSideEffect('Neptune#fts.sortBy', 'name.value')
```

- `score`: classificar por pontuação de correspondência (padrão).

  Se o parâmetro `sortOrder` estiver presente, mas `sortBy` não estiver presente, os resultados serão classificados por `score` na ordem especificada por `sortOrder`.
- `id`: classificar por ID, o que significa o URI de assunto do SPARQL ou o ID de borda ou vértice do Gremlin.

  Por exemplo, em uma consulta SPARQL, é possível especificar:

```
    neptune-fts:config neptune-fts:sortBy 'Neptune#fts.entity_id' .
```

  Em uma consulta Gremlin, é possível especificar:

```
    .withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.entity_id')
```

- `label`: ordenar por rótulo.

  Por exemplo, em uma consulta SPARQL, é possível especificar:

```
    neptune-fts:config neptune-fts:sortBy 'Neptune#fts.entity_type' .
```

  Em uma consulta Gremlin, é possível especificar:

```
    .withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.entity_type')
```

- `doc_type`: classificar por tipo de documento (ou seja, SPARQL ou Gremlin).

  Por exemplo, em uma consulta SPARQL, é possível especificar:

```
    neptune-fts:config neptune-fts:sortBy 'Neptune#fts.document_type' .
```

Em uma consulta Gremlin, é possível especificar:

```
.withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.document_type')
```

Por padrão, os resultados do OpenSearch não são classificados e a ordem não é determinística, o que significa que a mesma consulta pode exibir itens em uma ordem diferente a cada execução. Por esse motivo, se o resultado definido for maior que `max_result_window`, um subconjunto diferente dos resultados totais pode ser retornado a cada execução. No entanto, ao classificar você pode tornar os resultados de execuções diferentes mais comparáveis diretamente.

Se nenhum parâmetro `sortOrder` acompanhar o `sortBy`, será usada a ordem decrescente (`DESC`) do maior para o menor.

- **`sortOrder`**: um parâmetro opcional que permite especificar se os resultados do OpenSearch são classificados do menor para o maior ou do maior para o menor (padrão):

  - `ASC`: ordem crescente, do menor para o maior.
  - `DESC`: ordem decrescente, do maior para o menor.

    Este é o valor padrão usado quando o parâmetro `sortBy` for apresentado, mas nenhum `sortOrder` tiver sido especificado.

  Se nem `sortBy` nem `sortOrder` estiverem presentes, os resultados do OpenSearch não serão classificados por padrão.

Indexação sem string do OpenSearch no Amazon Neptune

A indexação do OpenSearch sem string no Amazon Neptune permite replicar valores que não sejam string para predicados no OpenSearch usando o instrumento de sondagem de fluxos. Todos os valores de predicados que podem ser convertidos com segurança em um mapeamento ou um tipo de dados correspondente do OpenSearch são então replicados no OpenSearch.

Para que a indexação sem string seja habilitada em uma nova pilha, o sinalizador `Enable Non-String Indexing` no modelo AWS CloudFormation deve ser definido como `true`. Essa é a configuração padrão. Para atualizar uma pilha existente com o objetivo de oferecer compatibilidade com a indexação sem string, veja Atualizar uma pilha existente abaixo.

> **Note**
>
> - É melhor não habilitar a indexação sem string em versões do mecanismo anteriores a **`1.0.4.2`**.
> - As consultas do OpenSearch que usam expressões regulares para nomes de campo correspondentes a vários campos, alguns dos quais contêm valores de string e outros que não contêm valores de string, falham com um erro. O mesmo acontecerá se as consultas de pesquisa de texto completo no Neptune forem desse tipo.
> - Ao classificar por um campo que não seja de string, acrescente “valor” ao nome do campo para diferenciá-lo de um campo de string.

Sumário

Atualizar uma pilha de pesquisa de texto completo existente do Neptune para oferecer compatibilidade com a indexação sem string

Se você já estiver usando a pesquisa de texto completo do Neptune, veja as etapas que você precisa seguir para oferecer compatibilidade com a indexação sem string:

1. Interrompa a função do Lambda do instrumento de sondagem de fluxos. Isso garante que nenhuma nova atualização seja copiada durante a exportação. Faça isso desabilitando a regra de evento de nuvem que invoca a função do Lambda:

   - No AWS Management Console, acesse o CloudWatch.
   - Selecione Regras.
   - Escolha a regra com o nome do instrumento de sondagem de fluxos do Lambda.
   - Selecione desabilitar para desabilitar temporariamente a regra.

2. Exclua o índice atual do Neptune no OpenSearch. Use a seguinte consulta `curl` para excluir o índice `amazon_neptune` do cluster do OpenSearch:

   ```
   curl -X DELETE "your OpenSearch endpoint/amazon_neptune"
   ```

3. Inicie uma exportação única do Neptune para o OpenSearch. É melhor configurar uma nova pilha do OpenSearch neste momento, para que novos artefatos sejam coletados para o instrumento de sondagem que executa a exportação.

   Siga as etapas listadas aqui no GitHub para iniciar a exportação única dos dados do Neptune para o OpenSearch.

4. Atualize os artefatos do Lambda para o instrumento de sondagem de fluxos existente. Depois que a exportação dos dados do Neptune para o OpenSearch for concluída com êxito, siga estas etapas:

   - No AWS Management Console, navegue até AWS CloudFormation.
   - Escolha a pilha AWS CloudFormation principal.
   - Selecione a opção Atualizar para a pilha.
   - Selecione Substituir modelo atual nas opções.
   - Na origem do modelo, selecione URL do Amazon S3.
   - Para o URL do Amazon S3, insira:

     ```
     https://aws-neptune-customer-samples.s3.amazonaws.com/neptune-stream/
     neptune_to_elastic_search.json
     ```

   - Selecione Próximo sem alterar nenhum parâmetro AWS CloudFormation.
   - Selecione Atualizar pilha. O AWS CloudFormation substituirá os artefatos do código do Lambda do instrumento de sondagem de fluxos pelos artefatos mais recentes.

5. Inicie o instrumento de sondagem de fluxos novamente. Faça isso habilitando a regra apropriada do CloudWatch:

   - No AWS Management Console, acesse o CloudWatch.
   - Selecione Regras.
   - Escolha a regra com o nome do instrumento de sondagem de fluxos do Lambda.
   - Selecione Habilitar.

Filtrar quais campos são indexados na pesquisa de texto completo do Neptune

Há dois campos nos detalhes do modelo do AWS CloudFormation que permitem especificar chaves de propriedade ou predicado ou tipos de dados a serem excluídos da indexação do OpenSearch:

Filtrar por nome de propriedade ou predicado

É possível usar o parâmetro opcional do modelo do AWS CloudFormation denominado `Properties to exclude from being inserted into Elastic Search Index` para fornecer uma lista delimitada por vírgulas de chaves de propriedade ou predicado a serem excluídas da indexação do OpenSearch.

Por exemplo, digamos que você defina esse parâmetro como `bob`.

```
"Properties to exclude from being inserted into Elastic Search Index" : bob
```

Nesse caso, o registro de fluxo da seguinte consulta de atualização do Gremlin seria descartado em vez de inserido no índice:

```
g.V("1").property("bob", "test")
```

Da mesma forma, é possível definir o parâmetro como `http://my/example#bob`:

```
"Properties to exclude from being inserted into Elastic Search Index" : http://my/
example#bob
```

Nesse caso, o registro de fluxo da seguinte consulta de atualização do SPARQL seria descartado em vez de inserido no índice:

```
PREFIX ex: <http://my/example#>
INSERT DATA { ex:s1 ex:bob "test"}.
```

Se você não inserir nada nesse parâmetro do modelo do AWS CloudFormation, todas as chaves de propriedade não excluídas de outra forma serão indexadas.

Filtrar por tipo de valor de propriedade ou predicado

É possível usar o parâmetro opcional do modelo do AWS CloudFormation denominado `Datatypes to exclude from being inserted into Elastic Search Index` para fornecer uma lista delimitada por vírgulas de tipos de dados de valor de propriedade ou predicado a serem excluídos da indexação do OpenSearch.

Para SPARQL, não é necessário listar o URI completo do tipo XSD. Basta listar o token do tipo de dados. Os tokens de tipos de dados válidos que você pode listar são:

- `string`
- `boolean`
- `float`
- `double`
- `dateTime`
- `date`
- `time`
- `byte`
- `short`
- `int`
- `long`
- `decimal`
- `integer`
- `nonNegativeInteger`
- `nonPositiveInteger`
- `negativeInteger`
- `unsignedByte`
- `unsignedShort`

- `unsignedInt`
- `unsignedLong`

Para Gremlin, os tipos de dados válidos para listar são:

- `string`
- `date`
- `bool`
- `byte`
- `short`
- `int`
- `long`
- `float`
- `double`

Por exemplo, digamos que você defina esse parâmetro como `string`.

```
"Datatypes to exclude from being inserted into Elastic Search Index" : string
```

Nesse caso, o registro de fluxo da seguinte consulta de atualização do Gremlin seria descartado em vez de inserido no índice:

```
g.V("1").property("myStringval", "testvalue")
```

Da mesma forma, é possível definir o parâmetro como `int`:

```
"Datatypes to exclude from being inserted into Elastic Search Index" : int
```

Nesse caso, o registro de fluxo da seguinte consulta de atualização do SPARQL seria descartado em vez de inserido no índice:

```
PREFIX ex: <http://my/example#>
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>
INSERT DATA { ex:s1 ex:bob "11"^^xsd:int }.
```

Se você não inserir nada nesse parâmetro do modelo do AWS CloudFormation, todas as propriedades cujos valores podem ser convertidos com segurança em equivalentes do OpenSearch serão indexadas. Os tipos listados que não são compatíveis com a linguagem de consulta são ignorados.

Mapeamento dos tipos de dados SPARQL e Gremlin para o OpenSearch

Novos mapeamentos de tipos de dados no OpenSearch são criados com base no tipo de dados que está sendo usado na propriedade ou no objeto. Como alguns campos contêm valores de tipos diferentes, o mapeamento inicial pode excluir alguns valores do campo.

Os tipos de dados do Neptune são associados aos tipos de dados do OpenSearch da seguinte forma:

Tipos do SPARQL	Tipos do Gremlin	Tipos do OpenSearch
`XSD:int`	`byte`	`long`
`XSD:unsignedInt`	`short`	
`XSD:integer`	`int`	
`XSD:byte`	`long`	
`XSD:unsignedByte`		
`XSD:short`		
`XSD:unsignedShort`		
`XSD:long`		
`XSD:unsignedLong`		
`XSD:float`	`float`	`double`
`XSD:double`	`double`	
`XSD:decimal`		
`XSD:boolean`	`bool`	`boolean`

Tipos do SPARQL	Tipos do Gremlin	Tipos do OpenSearch

`XSD:datetime`
 	
`XSD:date`
	`date`	`date`

`XSD:string`
 	
`XSD:time`
	`string`	`text`

Tipo de dados personalizado	N/D	`text`
Qualquer outro tipo de dados	N/D	`text`

Por exemplo, a seguinte consulta de atualização do Gremlin faz com que um novo mapeamento para “NewField” seja adicionado ao OpenSearch `{ "type" : "double" }`:

```
g.V("1").property("newField" 10.5)
```

Da mesma forma, a seguinte consulta de atualização do SPARQL faz com que um novo mapeamento para “ex:byte” seja adicionado ao OpenSearch `{ "type" : "long" }`:

```
PREFIX ex: <http://my/example#>
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>

INSERT DATA { ex:test ex:byte "123"^^xsd:byte }.
```

> **Note**
>
> Como você pode ver, um item associado do Neptune ao OpenSearch pode terminar com tipos de dados diferentes no OpenSearch e no Neptune. No entanto, há um campo de texto explícito no OpenSearch, “tipo de dados”, que registra o tipo de dados que o item tem no Neptune.

Validação de mapeamentos de dados

Os dados são replicados do Neptune no OpenSearch usando este processo:

- Se já houver um mapeamento para o campo em questão no OpenSearch:
  - Se os dados puderem ser convertidos com segurança no mapeamento existente usando regras de validação de dados, armazene o campo no OpenSearch.
  - Caso contrário, elimine o registro de atualização de fluxos correspondente.
- Se não houver mapeamento existente para o campo em questão, encontre um tipo de dados do OpenSearch correspondente ao tipo de dados do campo no Neptune.
  - Se os dados do campo puderem ser convertidos com segurança no tipo de dados do OpenSearch usando regras de validação de dados, armazene o novo mapeamento e os dados do campo no OpenSearch.
  - Caso contrário, elimine o registro de atualização de fluxos correspondente.

Os valores são validados em relação aos tipos equivalentes do OpenSearch ou aos mapeamentos existentes do OpenSearch e não aos tipos do Neptune. Por exemplo, a validação do valor `"123"` no `"123"^^xsd:int` é feita em relação ao tipo `long` e não ao tipo `int`.

Embora o Neptune tente replicar todos os dados no OpenSearch, há casos em que os tipos de dados no OpenSearch são totalmente diferentes dos tipos no Neptune e, nesses casos, os registros são ignorados em vez de serem indexados no OpenSearch.

Por exemplo, no Neptune, uma propriedade pode ter vários valores de tipos diferentes, enquanto no OpenSearch um campo deve ter o mesmo tipo em todo o índice.

Ao habilitar os logs de depuração, é possível ver quais registros foram descartados durante a exportação do Neptune para o OpenSearch. É um exemplo de entrada de log de depuração:

```
Dropping Record : Data type not a valid Gremlin type
<Record>
```

Os tipos de dados são validados da seguinte forma:

- **`text`**: todos os valores no Neptune podem ser associados com segurança ao texto no OpenSearch.
- **`long`**: as seguintes regras para os tipos de dados do Neptune se aplicam quando o tipo de mapeamento do OpenSearch é longo (nos exemplos abaixo, presume-se que `"testLong"` tenha um tipo de mapeamento `long`):
  - `boolean`: inválido, não pode ser convertido e o registro de atualização de fluxos correspondente é descartado.

São exemplos de Gremlin inválidos:

```
"testLong" : true.
"testLong" : false.
```

São exemplos de SPARQL inválidos:

```
":testLong" : "true"^^xsd:boolean
":testLong" : "false"^^xsd:boolean
```

- `datetime`: inválido, não pode ser convertido e o registro de atualização de fluxos correspondente é descartado.

  É um exemplo de Gremlin inválido:

  ```
  ":testLong" :  datetime('2018-11-04T00:00:00').
  ```

  É um exemplo de SPARQL inválido:

  ```
  ":testLong" : "2016-01-01"^^xsd:date
  ```

- `float`, `double` ou `decimal`: se o valor no Neptune for um número inteiro que caiba em 64 bits, ele será válido e será armazenado no OpenSearch como um valor longo, mas se tiver uma parte fracionária ou for `NaN` ou `INF`, ou for maior que 9.223.372.036.854.775.807 ou menor que -9.223.372.036.857.75.808, não será válido, e o registro de atualização de fluxos correspondente será descartado.

  São exemplos de Gremlin válidos:

  ```
  "testLong" :  145.0.
  ":testLong" :  123
  ":testLong" :  -9223372036854775807
  ```

  São exemplos de SPARQL válidos:

  ```
  ":testLong" : "145.0"^^xsd:float
  ":testLong" :  145.0
  ":testLong" : "145.0"^^xsd:double
  ":testLong" : "145.0"^^xsd:decimal
  ```

```
":testLong" : "-9223372036854775807"
```

São exemplos de Gremlin inválidos:

```
"testLong" :  123.45
":testLong" :  9223372036854775900
```

São exemplos de SPARQL inválidos:

```
":testLong" :  123.45
":testLong" :  9223372036854775900
":testLong" : "123.45"^^xsd:float
":testLong" : "123.45"^^xsd:double
":testLong" : "123.45"^^xsd:decimal
```

- `string`: se o valor no Neptune for uma representação em string de um valor inteiro que possa estar contido em um número inteiro de 64 bits, ele será válido e será convertido em um `long` no OpenSearch. Qualquer outro valor de string será inválido para um mapeamento `long` do ElasticSearch e o registro de atualização de fluxos correspondente será descartado.

  São exemplos de Gremlin válidos:

  ```
  "testLong" :  "123".
  ":testLong" :  "145.0"
  ":testLong" :  "-9223372036854775807"
  ```

  São exemplos de SPARQL válidos:

  ```
  ":testLong" : "145.0"^^xsd:string
  ":testLong" : "-9223372036854775807"^^xsd:string
  ```

  São exemplos de Gremlin inválidos:

  ```
  "testLong" :  "123.45"
  ":testLong" :  "9223372036854775900"
  ":testLong" :  "abc"
  ```

  São exemplos de SPARQL inválidos:

```
  ":testLong" : "123.45"^^xsd:string
  ":testLong" : "abc"
  ":testLong" : "9223372036854775900"^^xsd:string
```

- **`double`**: se o tipo de mapeamento do OpenSearch for `double`, as seguintes regras se aplicarão (aqui, presume-se que o campo “testDouble” tenha um mapeamento `double` no OpenSearch):
  - `boolean`: inválido, não pode ser convertido e o registro de atualização de fluxos correspondente é descartado.

    São exemplos de Gremlin inválidos:

    ```
      "testDouble" : true.
      "testDouble" : false.
    ```

    São exemplos de SPARQL inválidos:

    ```
      ":testDouble" : "true"^^xsd:boolean
      ":testDouble" : "false"^^xsd:boolean
    ```

  - `datetime`: inválido, não pode ser convertido e o registro de atualização de fluxos correspondente é descartado.

    É um exemplo de Gremlin inválido:

    ```
      ":testDouble" :  datetime('2018-11-04T00:00:00').
    ```

    É um exemplo de SPARQL inválido:

    ```
      ":testDouble" : "2016-01-01"^^xsd:date
    ```

  - Ponto flutuante `NaN` ou `INF`: se o valor em SPARQL for um ponto flutuante `NaN` ou `INF`, ele não será válido e o registro de atualização de fluxos correspondente será descartado.

    São exemplos de SPARQL inválidos:

    ```
    "  :testDouble" : "NaN"^^xsd:float
      ":testDouble" : "NaN"^^double
      ":testDouble" : "INF"^^double
      ":testDouble" : "-INF"^^double
    ```

- número ou string numérica: se o valor no Neptune for qualquer outro número ou representação de string numérica de um número que possa ser expresso com segurança como `double`, ele será válido e convertido em `double` no OpenSearch. Qualquer outro valor de string será inválido para um mapeamento `double` do OpenSearch e o registro de atualização de fluxos correspondente será descartado.

  São exemplos de Gremlin válidos:

  ```
  "testDouble" :  123
  ":testDouble" :  "123"
  ":testDouble" :  145.67
  ":testDouble" :  "145.67"
  ```

  São exemplos de SPARQL válidos:

  ```
  ":testDouble" :  123.45
  ":testDouble" :  145.0
  ":testDouble" : "123.45"^^xsd:float
  ":testDouble" : "123.45"^^xsd:double
  ":testDouble" : "123.45"^^xsd:decimal
  ":testDouble" : "123.45"^^xsd:string
  ```

  É um exemplo de Gremlin inválido:

  ```
  ":testDouble" :  "abc"
  ```

  É um exemplo de SPARQL inválido:

  ```
  ":testDouble" : "abc"
  ```

- **`date`**: se o tipo de mapeamento do OpenSearch for `date`, os valores `date` e `dateTime` do Neptune serão válidos, assim como qualquer valor de string que possa ser analisado com êxito em um formato `dateTime`.

  São exemplos válidos no Gremlin ou no SPARQL:

  ```
  Date(2016-01-01)
  "2016-01-01" "
  2003-09-25T10:49:41"
  ```

```
  "2003-09-25T10:49"
  "2003-09-25T10"
  "20030925T104941-0300"
  "20030925T104941"
  "2003-Sep-25" "
  Sep-25-2003"
  "2003.Sep.25"
  "2003/09/25"
  "2003 Sep 25" "
  Wed, July 10, '96"
  "Tuesday, April 12, 1952 AD 3:30:42pm PST"
  "123"
  "-123"
  "0"
  "-0"
  "123.00"
  "-123.00"
```

São exemplos inválidos:

```
  123.45
  True
  "abc"
```

Exemplos de consulta do OpenSearch sem string no Neptune

No momento, o Neptune não é compatível com consultas de intervalo do OpenSearch diretamente. No entanto, é possível obter o mesmo efeito usando a sintaxe Lucene e query-type="query_string", como você pode ver nos exemplos de consulta a seguir.

1. Obter todos os vértices com idade maior que 30 e nome começando com “Si”

No Gremlin:

```
g.withSideEffect('Neptune#fts.endpoint', 'http://your-es-endpoint')
 .withSideEffect("Neptune#fts.queryType", "query_string")
 .V().has('*', 'Neptune#fts predicates.age.value:>30 && predicates.name.value:Si*');
```

No SPARQL:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

```
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://localhost:9200' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query "predicates.\\*foaf\\*age.value:>30 AND
 predicates.\\*foaf\\*name.value:Si*" .
    neptune-fts:config neptune-fts:field '*' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Para resumir, aqui, "`\\*foaf\\*age` é usado em vez do URI completo. Essa expressão regular recuperará todos os campos que tenham `foaf` e `age` no URI.

2. Obter todos os nós com idades entre 10 e 50 e um nome com uma combinação difusa com “Ronka”

No Gremlin:

```
g.withSideEffect('Neptune#fts.endpoint', 'http://your-es-endpoint')
 .withSideEffect("Neptune#fts.queryType", "query_string")
 .V().has('*', 'Neptune#fts predicates.age.value:[10 TO 50] AND
 predicates.name.value:Ronka~');
```

No SPARQL:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://localhost:9200' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query "predicates.\\*foaf\\*age.value:[10 TO 50] AND
 predicates.\\*foaf\\*name.value:Ronka~" .
    neptune-fts:config neptune-fts:field '*' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

3. Obter todos os nós com um carimbo de data/hora que se enquadra nos últimos 25 dias

No Gremlin:

```
g.withSideEffect('Neptune#fts.endpoint', 'http://your-es-endpoint')
 .withSideEffect("Neptune#fts.queryType", "query_string")
 .V().has('*', 'Neptune#fts predicates.timestamp.value:>now-25d');
```

No SPARQL:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://localhost:9200' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query "predicates.\\*foaf\
\*timestamp.value:>now-25d~" .
    neptune-fts:config neptune-fts:field '*' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

4. Obter todos os nós com um carimbo de data/hora que se enquadre em um ano e mês específicos

No Gremlin, usar expressões matemáticas de data na sintaxe Lucene, para dezembro de 2020:

```
g.withSideEffect('Neptune#fts.endpoint', 'http://your-es-endpoint')
 .withSideEffect("Neptune#fts.queryType", "query_string")
 .V().has('*', 'Neptune#fts predicates.timestamp.value:>2020-12');
```

Uma alternativa ao Gremlin:

```
g.withSideEffect('Neptune#fts.endpoint', 'http://your-es-endpoint')
 .withSideEffect("Neptune#fts.queryType", "query_string")
 .V().has('*', 'Neptune#fts predicates.timestamp.value:[2020-12 TO 2021-01]');
```

Execução de consulta de pesquisa de texto completo no Amazon Neptune

Em uma consulta que inclua pesquisa de texto completo, o Neptune tenta colocar as chamadas de pesquisa de texto completo primeiro, antes de outras partes da consulta. Isso reduz o número de chamadas para o OpenSearch e, na maioria dos casos, melhora significativamente o desempenho. No entanto, essa não é de forma alguma uma regra rígida. Há situações, por exemplo, em que `PatternNode` ou `UnionNode` pode preceder uma chamada de pesquisa em texto completo.

Por exemplo, pense na seguinte consulta do Gremlin para um banco de dados que contenha cem mil instâncias de `Person`:

```
g.withSideEffect('Neptune#fts.endpoint', 'your-es-endpoint-URL')
 .hasLabel('Person')
 .has('name', 'Neptune#fts marcello~');
```

Se essa consulta fosse executada na ordem em que as etapas são exibidas, cem mil soluções fluiriam para o OpenSearch, causando centenas de chamadas do OpenSearch. Na verdade, o Neptune chama o OpenSearch primeiro e depois une os resultados aos resultados do Neptune. Na maioria dos casos, isso é muito mais rápido do que executar a consulta na ordem original.

É possível impedir essa reordenação da execução de etapas de consulta usando a dica de consulta noReordering :

```
g.withSideEffect('Neptune#fts.endpoint', 'your-es-endpoint-URL')
 .withSideEffect('Neptune#noReordering', true)
 .hasLabel('Person')
 .has('name', 'Neptune#fts marcello~');
```

Neste segundo caso, a etapa `.hasLabel` é executada primeiro e, logo depois, a etapa `.has('name', 'Neptune#fts marcello~')`.

Para outro exemplo, considere uma consulta SPARQL em relação ao mesmo tipo de dados:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT ?person WHERE {
  ?person rdf:type foaf:Person .
```

```
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:query 'mike' .
    neptune-fts:config neptune-fts:return ?person .
  }
}
```

Aqui, novamente, o Neptune executa a parte `SERVICE` da consulta primeiro e depois une os resultados aos dados `Person`. É possível suprimir esse comportamento usando a dica de consulta joinOrder:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT ?person WHERE {
  hint:Query hint:joinOrder "Ordered" .
  ?person rdf:type foaf:Person .
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:query 'mike' .
    neptune-fts:config neptune-fts:return ?person .
  }
}
```

Novamente, na segunda consulta, as partes são executadas na ordem em que são exibidas na consulta.

Exemplos de consulta do SPARQL usando a pesquisa de texto completo no Neptune

Veja alguns exemplos de consulta do SPARQL que usam a pesquisa de texto completo no Amazon Neptune.

Exemplo de consulta de correspondência do SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
```

```
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:queryType 'match' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:query 'michael' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Exemplo de consulta de prefixo do SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:queryType 'prefix' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:query 'mich' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Exemplo de consulta difusa do SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:queryType 'fuzzy' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:query 'mikael' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Exemplo de consulta de termo do SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

```
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:queryType 'term' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:query 'Dr. Kunal' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Exemplo de consulta query_string do SPARQL

Esta consulta especifica múltiplos campos.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ OR rondelli' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:field foaf:surname .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Exemplo de consulta simple_query_string do SPARQL

A consulta a seguir especifica os campos usando o caractere curinga (*).

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:queryType 'simple_query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ | rondelli' .
    neptune-fts:config neptune-fts:field '*' .
    neptune-fts:config neptune-fts:return ?res .
  }
```

```
}
```

Exemplo de consulta de campo de classificação por string do SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ | rondelli' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:sortOrder 'asc' .
    neptune-fts:config neptune-fts:sortBy foaf:name .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Exemplo de consulta de campo de classificação sem string do SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ | rondelli' .
    neptune-fts:config neptune-fts:field foaf:name.value .
    neptune-fts:config neptune-fts:sortOrder 'asc' .
    neptune-fts:config neptune-fts:sortBy dc:date.value .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Exemplo de consulta de classificação por ID do SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
```

```
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ | rondelli' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:sortOrder 'asc' .
    neptune-fts:config neptune-fts:sortBy 'Neptune#fts.entity_id' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Exemplo de consulta de classificação por rótulo do SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ | rondelli' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:sortOrder 'asc' .
    neptune-fts:config neptune-fts:sortBy 'Neptune#fts.entity_type' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Exemplo de consulta de classificação por doc_type do SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ | rondelli' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:sortOrder 'asc' .
    neptune-fts:config neptune-fts:sortBy 'Neptune#fts.document_type' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Exemplo de uso da sintaxe Lucene no SPARQL

A sintaxe Lucene só é compatível com consultas `query_string` no OpenSearch.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'predicates.\\foaf\\name.value:micheal AND
 predicates.\\foaf\\surname.value:sh' .
    neptune-fts:config neptune-fts:field '' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Usar a pesquisa de texto completo do Neptune em consultas do Gremlin

O `NeptuneSearchStep` permite consultas de pesquisa de texto completo para a parte de uma travessia do Gremlin que não é convertida em etapas do Neptune. Por exemplo, considere uma consulta como a seguinte.

```
g.withSideEffect("Neptune#fts.endpoint", "your-es-endpoint-URL")
  .V()
      .tail(100)
      .has("name", "Neptune#fts mark*")            <== # Limit the search on name
```

Esta consulta é convertida no percurso a seguir no Neptune.

```
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .],
 {estimatedCardinality=INFINITY}
        }, annotations={path=[Vertex(?1):GraphStep], maxVarId=4}
```

```
    },
    NeptuneTraverserConverterStep
]
+ not converted into Neptune steps: [NeptuneTailGlobalStep(100),
 NeptuneTinkerpopTraverserConverterStep, NeptuneSearchStep {
    JoinGroupNode {
        SearchNode[(idVar=?3, query=mark*, field=name) . project ask .],
 {endpoint=your-OpenSearch-endpoint-URL}
    }
    JoinGroupNode {
        SearchNode[(idVar=?3, query=mark*, field=name) . project ask .],
 {endpoint=your-OpenSearch-endpoint-URL}
    }
}]
```

Os exemplos a seguir são das consultas do Gremlin em relação a dados de rotas aéreas:

Consulta **`match`** sem distinção de maiúsculas ou minúsculas do Gremlin

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'match')
  .V().has("city","Neptune#fts dallas")

==>v[186]
==>v[8]
```

Consulta **`match`** do Gremlin

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'match')
  .V().has("city","Neptune#fts southampton")
     .local(values('code','city').fold())
     .limit(5)

==>[SOU, Southampton]
```

Consulta **`fuzzy`** do Gremlin

```
g.withSideEffect("Neptune#fts.endpoint",
```

```
                  "your-OpenSearch-endpoint-URL")
  .V().has("city","Neptune#fts allas~").values('city').limit(5)

==>Dallas
==>Dallas
==>Walla Walla
==>Velas
==>Altai
```

Consulta difusa **query_string** do Gremlin

```
g.withSideEffect("Neptune#fts.endpoint",
                  "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'query_string')
  .V().has("city","Neptune#fts allas~").values('city').limit(5)

==>Dallas
==>Dallas
```

Consulta de expressão regular **query_string** do Gremlin

```
g.withSideEffect("Neptune#fts.endpoint",
                  "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'query_string')
  .V().has("city","Neptune#fts /[dp]allas/").values('city').limit(5)

==>Dallas
==>Dallas
```

Consulta híbrida do Gremlin

Esta consulta usa um índice interno do Neptune e o índice do OpenSearch na mesma consulta.

```
g.withSideEffect("Neptune#fts.endpoint",
                  "your-OpenSearch-endpoint-URL")
  .V().has("region","GB-ENG")
      .has('city','Neptune#fts L*')
      .values('city')
      .dedup()
      .limit(10)
```

```
==>London
==>Leeds
==>Liverpool
==>Land's End
```

Exemplo de pesquisa de texto completo simples do Gremlin

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .V().has('desc','Neptune#fts regional municipal')
      .local(values('code','desc').fold())
      .limit(100)

==>[HYA, Barnstable Municipal Boardman Polando Field]
==>[SPS, Sheppard Air Force Base-Wichita Falls Municipal Airport]
==>[ABR, Aberdeen Regional Airport]
==>[SLK, Adirondack Regional Airport]
==>[BFD, Bradford Regional Airport]
==>[EAR, Kearney Regional Airport]
==>[ROT, Rotorua Regional Airport]
==>[YHD, Dryden Regional Airport]
==>[TEX, Telluride Regional Airport]
==>[WOL, Illawarra Regional Airport]
==>[TUP, Tupelo Regional Airport]
==>[COU, Columbia Regional Airport]
==>[MHK, Manhattan Regional Airport]
==>[BJI, Bemidji Regional Airport]
==>[HAS, Hail Regional Airport]
==>[ALO, Waterloo Regional Airport]
==>[SHV, Shreveport Regional Airport]
==>[ABI, Abilene Regional Airport]
==>[GIZ, Jizan Regional Airport]
==>[USA, Concord Regional Airport]
==>[JMS, Jamestown Regional Airport]
==>[COS, City of Colorado Springs Municipal Airport]
==>[PKB, Mid Ohio Valley Regional Airport]
```

Consulta do Gremlin usando **`query_string`** com operadores “+” e “-”

Embora o tipo de consulta `query_string` seja muito menos tolerante que o tipo `simple_query_string` padrão, ele não permite consultas mais precisas. A primeira consulta abaixo usa `query_string`, enquanto a segunda usa o padrão `simple_query_string`:

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'query_string')
 . V().has('desc','Neptune#fts +London -(Stansted|Gatwick)')
      .local(values('code','desc').fold())
      .limit(10)

==>[LHR, London Heathrow]
==>[YXU, London Airport]
==>[LTN, London Luton Airport]
==>[SEN, London Southend Airport]
==>[LCY, London City Airport]
```

Observe como `simple_query_string` nos exemplos abaixo ignora silenciosamente os operadores “+” e “-”:

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .V().has('desc','Neptune#fts +London -(Stansted|Gatwick)')
      .local(values('code','desc').fold())
      .limit(10)

==>[LHR, London Heathrow]
==>[YXU, London Airport]
==>[LGW, London Gatwick]
==>[STN, London Stansted Airport]
==>[LTN, London Luton Airport]
==>[SEN, London Southend Airport]
==>[LCY, London City Airport]
==>[SKG, Thessaloniki Macedonia International Airport]
==>[ADB, Adnan Menderes International Airport]
==>[BTV, Burlington International Airport]
```

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'query_string')
  .V().has('desc','Neptune#fts +(regional|municipal) -(international|bradford)')
      .local(values('code','desc').fold())
      .limit(10)

==>[CZH, Corozal Municipal Airport]
==>[MMU, Morristown Municipal Airport]
```

```
==>[YBR, Brandon Municipal Airport]
==>[RDD, Redding Municipal Airport]
==>[VIS, Visalia Municipal Airport]
==>[AIA, Alliance Municipal Airport]
==>[CDR, Chadron Municipal Airport]
==>[CVN, Clovis Municipal Airport]
==>[SDY, Sidney Richland Municipal Airport]
==>[SGU, St George Municipal Airport]
```

Consulte **query_string** do Gremlin com operadores **AND** e **OR**

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'query_string')
  .V().has('desc','Neptune#fts (St AND George) OR (St AND Augustin)')
      .local(values('code','desc').fold())
      .limit(10)

==>[YIF, St Augustin Airport]
==>[STG, St George Airport]
==>[SGO, St George Airport]
==>[SGU, St George Municipal Airport]
```

Consulta **term** do Gremlin

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'term')
  .V().has("SKU","Neptune#fts ABC123DEF9")
      .local(values('code','city').fold())
      .limit(5)

==>[AUS, Austin]
```

Consulta **prefix** do Gremlin

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'prefix')
  .V().has("icao","Neptune#fts ka")
```

```
      .local(values('code','icao','city').fold())
      .limit(5)

==>[AZO, KAZO, Kalamazoo]
==>[APN, KAPN, Alpena]
==>[ACK, KACK, Nantucket]
==>[ALO, KALO, Waterloo]
==>[ABI, KABI, Abilene]
```

Usar a sintaxe Lucene em Gremlin no Neptune

No Gremlin no Neptune, também é possível gravar consultas muito avançadas usando a sintaxe de consulta Lucene. Observe que a sintaxe Lucene só é compatível com consultas `query_string` no OpenSearch.

Suponha os seguintes dados:

```
g.addV("person")
        .property(T.id, "p1")
        .property("name", "simone")
        .property("surname", "rondelli")

g.addV("person")
        .property(T.id, "p2")
        .property("name", "simone")
        .property("surname", "sengupta")

g.addV("developer")
        .property(T.id, "p3")
        .property("name", "simone")
        .property("surname", "rondelli")
```

Usando a sintaxe Lucene, que é invocada quando `queryType` for `query_string`, é possível pesquisar esses dados por nome e sobrenome da seguinte forma:

```
g.withSideEffect("Neptune#fts.endpoint", "es_endpoint")
    .withSideEffect("Neptune#fts.queryType", "query_string")
    .V()
    .has("*", "Neptune#fts predicates.name.value:simone AND
 predicates.surname.value:rondelli")
```

```
==> v[p1], v[p3]
```

Observe que na etapa `has()` acima, o campo é substituído por `"*"`). Na verdade, qualquer valor inserido aqui será substituído pelos campos que você acessar na consulta. Acesse o campo do nome usando `predicates.name.value,` porque é como o modelo de dados é estruturado.

É possível pesquisar por nome, sobrenome e rótulo da seguinte forma:

```
g.withSideEffect("Neptune#fts.endpoint", getEsEndpoint())
    .withSideEffect("Neptune#fts.queryType", "query_string")
    .V()
    .has("*", "Neptune#fts predicates.name.value:simone AND
 predicates.surname.value:rondelli AND entity_type:person")

==> v[p1]
```

O rótulo é acessado usando `entity_type`, novamente porque é como o modelo de dados é estruturado.

Também é possível incluir condições de aninhamento:

```
g.withSideEffect("Neptune#fts.endpoint", getEsEndpoint())
    .withSideEffect("Neptune#fts.queryType", "query_string")
    .V()
    .has("*", "Neptune#fts (predicates.name.value:simone AND
 predicates.surname.value:rondelli AND entity_type:person) OR
 predicates.surname.value:sengupta")

==> v[p1], v[p2]
```

Inserir um grafo moderno do TinkerPop

```
g.addV('person').property(T.id, '1').property('name', 'marko').property('age', 29)
 .addV('personr').property(T.id, '2').property('name', 'vadas').property('age', 27)
 .addV('software').property(T.id, '3').property('name', 'lop').property('lang', 'java')
 .addV('person').property(T.id, '4').property('name', 'josh').property('age', 32)
 .addV('software').property(T.id, '5').property('name', 'ripple').property('lang',
 'java')
 .addV('person').property(T.id, '6').property('name', 'peter').property('age', 35)

g.V('1').as('a').V('2').as('b').addE('knows').from('a').to('b').property('weight',
 0.5f).property(T.id, '7')
```

```
 .V('1').as('a').V('3').as('b').addE('created').from('a').to('b').property('weight',
 0.4f).property(T.id, '9')
 .V('4').as('a').V('3').as('b').addE('created').from('a').to('b').property('weight',
 0.4f).property(T.id, '11')
 .V('4').as('a').V('5').as('b').addE('created').from('a').to('b').property('weight',
 1.0f).property(T.id, '10')
 .V('6').as('a').V('3').as('b').addE('created').from('a').to('b').property('weight',
 0.2f).property(T.id, '12')
 .V('1').as('a').V('4').as('b').addE('knows').from('a').to('b').property('weight',
 1.0f).property(T.id, '8')
```

Exemplo de valor do campo de classificação por string

```
g.withSideEffect("Neptune#fts.endpoint", "your-OpenSearch-endpoint-URL")
 .withSideEffect('Neptune#fts.queryType', 'query_string')
 .withSideEffect('Neptune#fts.sortOrder', 'asc')
 .withSideEffect('Neptune#fts.sortBy', 'name')
 .V().has('name', 'Neptune#fts marko OR vadas OR ripple')
```

Exemplo de valor do campo de classificação sem string

```
g.withSideEffect("Neptune#fts.endpoint", "your-OpenSearch-endpoint-URL")
 .withSideEffect('Neptune#fts.queryType', 'query_string')
 .withSideEffect('Neptune#fts.sortOrder', 'asc')
 .withSideEffect('Neptune#fts.sortBy', 'age.value')
 .V().has('name', 'Neptune#fts marko OR vadas OR ripple')
```

Exemplo de valor do campo de classificação por ID

```
g.withSideEffect("Neptune#fts.endpoint", "your-OpenSearch-endpoint-URL")
.withSideEffect('Neptune#fts.queryType', 'query_string')
.withSideEffect('Neptune#fts.sortOrder', 'asc')
.withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.entity_id')
.V().has('name', 'Neptune#fts marko OR vadas OR ripple')
```

Exemplo de valor do campo de classificação por rótulo

```
g.withSideEffect("Neptune#fts.endpoint", "your-OpenSearch-endpoint-URL")
 .withSideEffect('Neptune#fts.queryType', 'query_string')
```

```
  .withSideEffect('Neptune#fts.sortOrder', 'asc')
  .withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.entity_type')
  .V().has('name', 'Neptune#fts marko OR vadas OR ripple')
```

Exemplo de valor do campo de classificação por **`document_type`**

```
g.withSideEffect("Neptune#fts.endpoint", "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'query_string')
  .withSideEffect('Neptune#fts.sortOrder', 'asc')
  .withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.document_type')
  .V().has('name', 'Neptune#fts marko OR vadas OR ripple')
```

Solução de problemas da pesquisa de texto completo do Neptune

> **Note**
>
> Se você habilitou o controle de acesso refinado no cluster do OpenSearch, também precisará habilitar a autenticação do IAM no banco de dados Neptune.

Para diagnosticar problemas com a replicação do Neptune para o OpenSearch, consulte o CloudWatch Logs para a função do Lambda do instrumento de sondagem. Esses logs fornecem detalhes sobre o número de registros lidos no fluxo e o número de registros replicados com êxito no OpenSearch.

Também é possível alterar o nível de LOGGING para a função do Lambda ao alterar a variável de ambiente `LoggingLevel`.

> **Note**
>
> Com `LoggingLevel` definido como `DEBUG`, é possível visualizar detalhes adicionais, como registros de fluxo descartados e o motivo pelo qual cada um foi descartado, enquanto replica dados pelo StreamPoller do Neptune para o OpenSearch. Isso poderá ser útil se você descobrir que faltam registros.

A aplicação de consumidor de fluxos do Neptune publica duas métricas no CloudWatch que também podem ajudar a diagnosticar problemas:

- `StreamRecordsProcessed`: o número de registros processados pela aplicação por unidade de tempo. Útil no monitoramento da taxa de execução do aplicativo.
- `StreamLagTime`: a diferença de tempo em milissegundos entre a hora atual e a hora de confirmação de um registro de fluxo que está sendo processado. Esta métrica mostra o atraso do aplicativo de consumidor.

Além disso, todas as métricas relacionadas ao processo de replicação são expostas em um painel no CloudWatch com o mesmo nome `ApplicationName` fornecido ao instanciar a aplicação usando o modelo do CloudWatch.

Também é possível criar um alarme do CloudWatch que seja acionado sempre que a sondagem falhar mais de duas vezes seguidas. Faça isso ao definir o campo `CreateCloudWatchAlarm` como `true` ao instanciar o aplicativo. Depois especifique os endereços de e-mail que deseja notificar quando o alarme for acionado.

Solução de problemas de um processo que falha ao ler registros do fluxo

Se um processo falhar ao ler registros do fluxo, certifique-se de que:

- O fluxo esteja habilitado no cluster.
- O endpoint do fluxo do Neptune esteja no formato correto:
  - Para Gremlin ou openCypher: `https://`*`your cluster endpoint`*`:`*`your cluster port`*`/propertygraph/stream` ou o alias, `https://`*`your cluster endpoint`*`:`*`your cluster port`*`/pg/stream`.
  - Para o SPARQL: `https://`*`your cluster endpoint`*`:`*`your cluster port`*`/sparql/stream`.
- O endpoint do DynamoDB esteja configurado para a VPC.
- O monitoramento do endpoint está configurado para as sub-redes da VPC.

Solução de problemas de um processo que falha ao gravar dados no OpenSearch

Se um processo falhar ao gravar registros no OpenSearch, certifique-se de que:

- Sua versão do Elasticsearch seja 7.1 ou posterior, ou Opensearch 2.3 e posterior.

- O OpenSearch possa ser acessado pela função do Lambda do instrumento de sondagem na VPC.
- A política de segurança vinculada ao OpenSearch permita a entrada de solicitações HTTP/HTTPS de entrada.

Corrigir problemas fora de sincronia entre o Neptune e o OpenSearch em uma configuração de replicação existente

É possível usar as etapas abaixo para sincronizar o banco de dados Neptune e o domínio do OpenSearch com os dados mais recentes em caso de problemas de falta de sincronia entre eles resultantes de uma corrupção de dados ou `ExpiredStreamException`.

Observe que essa abordagem exclui todos os dados no domínio do OpenSearch e os sincroniza novamente do estado atual do banco de dados Neptune, portanto, nenhum dado precisa ser recarregado no banco de dados Neptune.

1. Desabilite o processo de replicação conforme descrito em Disabling (pausing) the stream poller process.
2. Exclua o índice do Neptune no domínio do OpenSearch usando o seguinte comando:

   ```
   curl -X DELETE "(your OpenSearch endpoint)/amazon_neptune"
   ```

3. Criar um clone do banco de dados (consulte Database Cloning in Neptune).
4. Obtenha o `eventID` mais recente dos fluxos no banco de dados clonado executando um comando desse tipo no endpoint da API de fluxos (consulte Calling the Neptune Streams REST API para obter mais informações):

   ```
   curl "https://(your neptune endpoint):(port)/(propertygraph or sparql)/stream?
   iteratorType=LATEST"
   ```

   Anote os valores nos campos `commitNum` e `opNum` no objeto `lastEventId` na resposta.
5. Use a ferramenta export-neptune-to-elasticsearch no github para realizar uma sincronização única do banco de dados clonado para o domínio do OpenSearch.
6. Acesse a tabela do DynamoDB para ver a pilha de replicação. O nome da tabela será o Nome da aplicação que você especificou no modelo do AWS CloudFormation (o padrão é `NeptuneStream`) com um sufixo `-LeaseTable`. Em outras palavras, o nome padrão da tabela é `NeptuneStream-LeaseTable`.

É possível examinar as linhas da tabela com a verificação porque só deve haver uma linha na tabela. Faça as seguintes alterações usando os valores `commitNum` e `opNum` que você registrou acima:

- Altere o valor do campo `checkpoint` na tabela para o valor que você anotou para `commitNum`.
- Altere o valor do campo `checkpointSubSequenceNumber` na tabela para o valor que você anotou para `opNum`.

7. Reabilite o processo de replicação conforme descrito em Re-enabling the stream poller process.
8. Exclua o banco de dados clonado e a pilha AWS CloudFormation criada para a ferramenta `export-neptune-to-elasticsearch`.

Usar funções do AWS Lambda no Amazon Neptune

As funções do AWS Lambda têm muitos usos nas aplicações Amazon Neptune. Aqui fornecemos orientações gerais para usar as funções do Lambda com qualquer um dos drivers e variantes de linguagem populares do Gremlin, bem como exemplos específicos de funções do Lambda escritas em Java, JavaScript e Python.

> **Note**
>
> A melhor maneira de usar as funções do Lambda com o Neptune mudou com as versões recentes do mecanismo. O Neptune costumava deixar conexões ociosas abertas muito depois de um contexto de execução do Lambda ter sido reciclado, podendo causar um vazamento de recursos no servidor. Para atenuar isso, costumávamos recomendar abrir e fechar uma conexão com cada invocação do Lambda. A partir da versão 1.0.3.0 do mecanismo, no entanto, o tempo limite da conexão ociosa foi reduzido para que as conexões não vazem mais após a reciclagem de um contexto de execução inativo do Lambda. Por isso, agora recomendamos o uso de uma única conexão durante o contexto de execução. Isso deve incluir algum tratamento de erros e código clichê de recuo e repetição para lidar com conexões fechadas inesperadamente.

Gerenciar conexões do WebSocket do Gremlin em funções do AWS Lambda

Se você usar uma variante da linguagem Gremlin para consultar o Neptune, o driver se conectará ao banco de dados usando uma conexão do WebSocket. Os WebSockets são projetados para ser compatíveis com cenários de conexão cliente-servidor de longa duração. O AWS Lambda, por outro lado, foi projetado para oferecer compatibilidade com execuções relativamente curtas e sem estado. Essa incompatibilidade na filosofia de design pode causar alguns problemas inesperados ao usar o Lambda para consultar o Neptune.

Uma função do AWS Lambda é executada em um contexto de execução que isola a função de outras funções. O contexto de execução é criado na primeira vez que a função é invocada e pode ser reutilizado para invocações subsequentes da mesma função.

No entanto, qualquer contexto de execução nunca é usado para lidar com várias invocações simultâneas da função. Se a função for invocada simultaneamente por vários clientes, o Lambda

criará um contexto de execução adicional para cada instância da função. Todos esses novos contextos de execução podem, por sua vez, ser reutilizados para invocações subsequentes da função.

Em algum momento, o Lambda recicla contextos de execução, principalmente se eles permanecerem inativos por algum tempo. O AWS Lambda expõe o ciclo de vida do contexto de execução, incluindo as fases `Init`, `Invoke` e `Shutdown` por meio de extensões do Lambda. Usando essas extensões, você pode escrever um código que limpe recursos externos, como conexões de banco de dados, quando o contexto de execução é reciclado.

Uma prática recomendada comum é abrir a conexão do banco de dados fora da função de manipulador do Lambda para que ela possa ser reutilizada com cada chamada do manipulador. Se a conexão com o banco de dados cair em algum momento, você poderá se reconectar de dentro do manipulador. No entanto, com essa abordagem, existe o perigo de vazamentos de conexão. Se uma conexão ociosa permanecer aberta por muito tempo após a destruição de um contexto de execução, cenários de invocação do Lambda intermitentes poderão gradualmente vazar conexões e esgotar os recursos do banco de dados.

Os limites de conexão e os tempos limite de conexão do Neptune mudaram com as versões mais recentes do mecanismo. Anteriormente, cada instância suportava até sessenta mil conexões do WebSocket. Agora, o número máximo de conexões simultâneas do WebSocket por instância do Neptune varia de acordo com o tipo de instância.

Além disso, a partir da versão 1.0.3.0 do mecanismo, o Neptune reduziu o tempo limite de inatividade das conexões de uma hora para cerca de vinte minutos. Se um cliente não fechar uma conexão, ela será fechada automaticamente após um tempo limite de inatividade de 20 a 25 minutos. O AWS Lambda não documenta os tempos de vida do contexto de execução, mas experiências demonstram que o novo tempo limite de conexão do Neptune se alinha bem com os tempos limite do contexto de execução inativo do Lambda. No momento em que um contexto de execução inativo é reciclado, há uma boa chance de a conexão já ter sido fechada pelo Neptune ou ser fechada logo depois.

Recomendações para uso do AWS Lambda com o Gremlin no Amazon Neptune

Agora, recomendamos usar uma única fonte de conexão e percurso de grafos durante toda a vida útil de um contexto de execução do Lambda, em vez de uma para cada invocação de função (cada invocação de função processa somente uma solicitação do cliente). Como as solicitações

simultâneas do cliente são tratadas por diferentes instâncias de função executadas em contextos de execução separados, não há necessidade de manter um grupo de conexões para lidar com solicitações simultâneas dentro de uma instância de função. Se o driver do Gremlin que você está usando tiver um grupo de conexões, configure-o para usar apenas uma conexão.

Para lidar com falhas de conexão, use a lógica de repetição em cada consulta. Embora o objetivo seja manter uma única conexão durante a vida útil de um contexto de execução, eventos de rede inesperados podem fazer com que essa conexão seja encerrada abruptamente. Essas falhas de conexão se manifestam como erros diferentes, dependendo do driver que você está usando. É necessário codificar a função do Lambda para lidar com esses problemas de conexão e tentar uma reconexão, se necessário.

Alguns drivers do Gremlin lidam automaticamente com as reconexões. O driver do Java, por exemplo, tenta automaticamente restabelecer a conectividade com o Neptune em nome do código cliente. Com esse driver, o código de função só precisa recuar e repetir a consulta. Os drivers do JavaScript e do Python, por outro lado, não implementam nenhuma lógica de reconexão automática, portanto, com esses drivers, o código de função deve tentar se reconectar depois de recuar e só repetir a consulta depois que a conexão for restabelecida.

Os exemplos de código aqui incluem a lógica de reconexão, em vez de presumir que o cliente esteja cuidando dela.

Recomendações para usar solicitações de gravação do Gremlin no Lambda

Se a função do Lambda modificar dados de grafos, pense em adotar uma estratégia de recuo e repetição para lidar com as seguintes exceções:

- **`ConcurrentModificationException`**: a semântica da transação do Neptune significa que as solicitações de gravação às vezes falham com uma `ConcurrentModificationException`. Nessas situações, experimente um mecanismo de repetição baseado em recuo exponencial.
- **`ReadOnlyViolationException`**: como a topologia do cluster pode mudar a qualquer momento como resultado de eventos planejados ou não, as responsabilidades de gravação podem migrar de uma instância no cluster para outra. Se o código de função tentar enviar uma solicitação de gravação a uma instância que não seja mais a principal (de gravador), a solicitação falhará com uma `ReadOnlyViolationException`. Quando isso acontecer, feche a conexão existente, reconecte-se ao endpoint do cluster e repita a solicitação.

Além disso, se você usar uma estratégia de recuo e repetição para lidar com problemas de solicitação de gravação, considere implementar consultas idempotentes para solicitações de criação e atualização (por exemplo, usando fold().coalesce().unfold().

Recomendações para usar solicitações de leitura do Gremlin no Lambda

Se você tiver uma ou mais réplicas de leitura no cluster, é uma boa ideia equilibrar as solicitações de leitura entre essas réplicas. Uma opção é usar o endpoint de leitor. O endpoint de leitor equilibra as conexões entre as réplicas, mesmo que a topologia do cluster mude quando você adiciona ou remove réplicas ou promove uma réplica para se tornar a nova instância principal.

No entanto, o uso do endpoint de leitor pode ocasionar um uso desigual dos recursos do cluster em algumas circunstâncias. O endpoint de leitor funciona periodicamente alterando o host para o qual a entrada DNS aponta. Se um cliente abrir muitas conexões antes que a entrada de DNS seja alterada, todas as solicitações de conexão serão enviadas a uma única instância do Neptune. Esse pode ser o caso de um cenário do Lambda de alto throughput em que um grande número de solicitações simultâneas para a função do Lambda faz com que vários contextos de execução sejam criados, cada um com a própria conexão. Se todas essas conexões forem criadas quase simultaneamente, é provável que todas apontem para a mesma réplica no cluster e permaneçam apontando para essa réplica até que os contextos de execução sejam reciclados.

Uma forma de distribuir solicitações entre instâncias é configurar a função do Lambda para se conectar a um endpoint de instância, escolhido aleatoriamente em uma lista de endpoints de instância de réplica, em vez do endpoint de leitor. A desvantagem dessa abordagem é que ela exige que o código do Lambda manipule as alterações na topologia do cluster, monitorando o cluster e atualizando a lista de endpoints sempre que a associação ao cluster mudar.

Se você estiver escrevendo uma função do Lambda em Java que precise equilibrar as solicitações de leitura entre instâncias no cluster, poderá usar o cliente do Gremlin para Amazon Neptune, um cliente do Gremlin em Java que conhece a topologia do cluster e que distribui de forma justa as conexões e as solicitações em um conjunto de instâncias em um cluster do Neptune. Esta postagem no blog inclui um exemplo de função do Lambda em Java que usa o cliente do Gremlin para Amazon Neptune.

Fatores que podem retardar o início a frio das funções do Lambda em Gremlin no Neptune

A primeira vez que uma função do AWS Lambda é invocada é chamada de inicialização a frio. Há vários fatores que podem aumentar a latência de uma inicialização a frio:

- Atribua memória suficiente à função do Lambda. A compilação durante uma inicialização a frio pode ser significativamente mais lenta para uma função do Lambda do que seria no EC2 porque o AWS Lambda aloca ciclos de CPU linearmente em proporção à memória atribuída à função. Com 1.792 MB de memória, uma função recebe o equivalente a uma vCPU inteira (um segundo de vCPU de créditos por segundo). O impacto de não atribuir memória suficiente para receber ciclos de CPU adequados é particularmente pronunciado para grandes funções do Lambda escritas em Java.
- Esteja ciente de que habilitar a autenticação do banco de dados do IAM pode retardar uma inicialização a frio: a autenticação do banco de dados do AWS Identity and Access Management (IAM) também poderá retardar as inicializações a frio, especialmente se a função do Lambda precisar gerar uma nova chave de assinatura. Essa latência afeta apenas a inicialização a frio e não as solicitações subsequentes, porque depois que a autenticação do banco de dados do IAM estabelece as credenciais de conexão, o Neptune só confirma periodicamente que elas ainda são válidas.

Usar o AWS CloudFormation para criar uma função do Lambda a ser usada no Neptune

É possível usar um modelo do AWS CloudFormation para criar uma função do AWS Lambda que pode acessar o Neptune.

1. Para iniciar a pilha de funções do Lambda no console do AWS CloudFormation, selecione um dos botões Iniciar pilha na tabela a seguir.

Região	Visualização	Visualizar no Designer	Executar
Leste dos EUA (Norte da Virgínia)	Visão	Visualizar no Designer	Launch Stack
Leste dos EUA (Ohio)	Visão	Visualizar no Designer	Launch Stack
Oeste dos EUA (N. da Califórnia)	Visão	Visualizar no Designer	Launch Stack

Região	Visualização	Visualizar no Designer	Executar
Oeste dos EUA (Oregon)	Visão	Visualizar no Designer	
Canadá (Central)	Visão	Visualizar no Designer	
América do Sul (São Paulo)	Visão	Visualizar no Designer	
Europa (Estocolmo)	Visão	Visualizar no Designer	
Europa (Irlanda)	Visão	Visualizar no Designer	
Europa (Londres)	Visão	Visualizar no Designer	
Europa (Paris)	Visão	Visualizar no Designer	
Europa (Frankfurt)	Visão	Visualizar no Designer	
Oriente Médio (Barém)	Visão	Visualizar no Designer	
Oriente Médio (Emirados Árabes Unidos)	Visão	Visualizar no Designer	
Israel (Tel Aviv)	Visão	Visualizar no Designer	
África (Cidade do Cabo)	Visão	Visualizar no Designer	

Região	Visualização	Visualizar no Designer	Executar
Ásia-Pacífico (Hong Kong)	Visão	Visualizar no Designer	
Ásia-Pacífico (Tóquio)	Visão	Visualizar no Designer	
Ásia-Pacífico (Seul)	Visão	Visualizar no Designer	
Ásia-Pacífico (Singapura)	Visão	Visualizar no Designer	
Ásia-Pacífico (Sydney)	Visão	Visualizar no Designer	
Ásia-Pacífico (Mumbai)	Visão	Visualizar no Designer	
China (Pequim)	Visão	Visualizar no Designer	
China (Ningxia)	Visão	Visualizar no Designer	
AWS GovCloud (Oeste dos EUA)	Visão	Visualizar no Designer	
AWS GovCloud (Leste dos EUA)	Visão	Visualizar no Designer	

2. Na página Select Template, escolha Next.
3. Na página Specify Details (Especificar detalhes), defina as seguintes opções:

   a. Escolha o runtime do Lambda dependendo do que você deseja usar na função do Lambda. Esses modelos do AWS CloudFormation atualmente oferecem suporte às seguintes linguagens:

- Python 3.9 (é associado ao `python39` no URL do Amazon S3).
- NodeJS 18 (é associado ao `nodejs18x` no URL do Amazon S3).
- Ruby 2.5 (é associado ao `ruby25` no URL do Amazon S3).

b. Forneça o endpoint do cluster do Neptune e o número da porta apropriados.

c. Forneça o grupo de segurança apropriado do Neptune.

d. Forneça os parâmetros apropriados de sub-rede do Neptune.

4. Escolha Next (Próximo).
5. Na página Options (Opções), escolha Next (Avançar).
6. Na página Revisar, marque a primeira caixa de seleção para confirmar que o AWS CloudFormation criará recursos do IAM.

   Em seguida, selecione Criar.

Se você precisar fazer suas próprias alterações no runtime do Lambda, baixe uma genérica de um local do Amazon S3 em sua região:

```
https://s3.Amazon region.amazonaws.com/aws-neptune-customer-samples-Amazon region/
lambda/runtime-language/lambda_function.zip.
```

Por exemplo:

```
https://s3.us-west-2.amazonaws.com/aws-neptune-customer-samples-us-west-2/lambda/
python36/lambda_function.zip
```

Exemplos de função do AWS Lambda para Amazon Neptune

Os exemplos de função do AWS Lambda a seguir, escritos em Java, JavaScript e Python, ilustram a aplicação de upsert a um único vértice com um ID gerado aleatoriamente com a expressão `fold().coalesce().unfold()`.

Grande parte do código em cada função é código clichê responsável por gerenciar e repetir conexões e consultas em caso de erro. A lógica real da aplicação e a consulta do Gremlin são implementadas nos métodos `doQuery()` e `query()`, respectivamente. Se você usar esses exemplos como base para as próprias funções do Lambda, poderá se concentrar em modificar `doQuery()` e `query()`.

As funções são configuradas para repetir consultas com falha cinco vezes, aguardando um segundo entre as tentativas.

As funções exigem que os valores estejam presentes nas seguintes variáveis de ambiente do Lambda:

- **`NEPTUNE_ENDPOINT`**: o endpoint de cluster de banco de dados do Neptune. Para Python, deve ser `neptuneEndpoint`.
- **`NEPTUNE_PORT`**: a porta do Neptune. Para Python, deve ser `neptunePort`.
- **`USE_IAM`** : (`true` ou `false`) se o banco de dados tiver a autenticação de banco de dados do AWS Identity and Access Management (IAM) habilitada, defina a variável de ambiente `USE_IAM` como `true`. Isso faz com que a função do Lambda assine por Sigv4 solicitações de conexão com o Neptune. Para essas solicitações de autenticação de banco de dados do IAM, garanta que a função de execução da função do Lambda tenha uma política do IAM apropriada anexada que permita que a função se conecte ao cluster de banco de dados do Neptune (consulte Tipos de política do IAM).

Exemplo de função do Lambda em Java para Amazon Neptune

Veja algumas considerações sobre funções do AWS Lambda em Java:

- O driver Java mantém o próprio grupo de conexões, que não é necessário, então configure o objeto `Cluster` com `minConnectionPoolSize(1)` e `maxConnectionPoolSize(1)`.
- O objeto `Cluster` pode demorar porque cria um ou mais serializadores (Gyro por padrão, além de outro, se você o tiver configurado para formatos de saída adicionais, como `binary`). Eles podem demorar um pouco para ser instanciados.
- O grupo de conexões é inicializado com a primeira solicitação. Nesse ponto, o driver configura a pilha `Netty`, aloca buffers de bytes e cria uma chave de assinatura caso você esteja usando a autenticação de banco de dados do IAM. Tudo isso pode aumentar a latência de inicialização a frio.
- O grupo de conexões do driver Java monitora a disponibilidade dos hosts do servidor e tenta se reconectar automaticamente em caso de falha da conexão. Ele inicia uma tarefa em segundo plano para tentar restabelecer a conexão. Use `reconnectInterval( )` para configurar o intervalo entre as tentativas de reconexão. Enquanto o driver está tentando se reconectar, a função do Lambda pode simplesmente repetir a consulta.

Se o intervalo entre as tentativas for menor do que o intervalo entre as tentativas de reconexão, ocorrerá novamente uma falha nas novas tentativas em uma conexão com falha porque o host será considerado indisponível. Isso não se aplica às novas tentativas de uma `ConcurrentModificationException`.

- Use o Java 8 em vez do Java 11. As otimizações do `Netty` não estão habilitadas por padrão no Java 11.
- Este exemplo usa Retry4j para novas tentativas.
- Para usar o driver de assinatura `Sigv4` na função do Lambda em Java, consulte os requisitos de dependência em Estabelecer conexão com o Neptune usando Java e Gremlin com a assinatura do Signature versão 4.

> **Warning**
>
> O `CallExecutor` do Retry4j pode não ser livre de threads. Pense em fazer com que cada thread use a própria instância `CallExecutor`.

```
package com.amazonaws.examples.social;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestStreamHandler;
import com.evanlennick.retry4j.CallExecutor;
import com.evanlennick.retry4j.CallExecutorBuilder;
import com.evanlennick.retry4j.Status;
import com.evanlennick.retry4j.config.RetryConfig;
import com.evanlennick.retry4j.config.RetryConfigBuilder;
import org.apache.tinkerpop.gremlin.driver.Cluster;
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.neptune.auth.NeptuneNettyHttpSigV4Signer;
import org.apache.tinkerpop.gremlin.driver.remote.DriverRemoteConnection;
import org.apache.tinkerpop.gremlin.driver.ser.Serializers;
import org.apache.tinkerpop.gremlin.process.traversal.AnonymousTraversalSource;
import org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversalSource;
import org.apache.tinkerpop.gremlin.structure.T;

import java.io.*;
import java.time.temporal.ChronoUnit;
import java.util.HashMap;
```

```
import java.util.Map;
import java.util.Random;
import java.util.concurrent.Callable;
import java.util.function.Function;

import static java.nio.charset.StandardCharsets.UTF_8;
import static org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.__.addV;
import static org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.__.unfold;

public class MyHandler implements RequestStreamHandler {

  private final GraphTraversalSource g;
  private final CallExecutor<Object> executor;
  private final Random idGenerator = new Random();

  public MyHandler() {

    this.g = AnonymousTraversalSource
            .traversal()
            .withRemote(DriverRemoteConnection.using(createCluster()));


    this.executor = new CallExecutorBuilder<Object>()
            .config(createRetryConfig())
            .build();

  }

  @Override
  public void handleRequest(InputStream input,
                            OutputStream output,
                            Context context) throws IOException {

    doQuery(input, output);
  }

  private void doQuery(InputStream input, OutputStream output) throws IOException {
    try {

      Map<String, Object> args = new HashMap<>();
      args.put("id", idGenerator.nextInt());

      String result = query(args);
```

```
      try (Writer writer = new BufferedWriter(new OutputStreamWriter(output, UTF_8))) {
          writer.write(result);
      }

    } finally {
        input.close();
        output.close();
    }
  }

  private String query(Map<String, Object> args) {
    int id = (int) args.get("id");

    @SuppressWarnings("unchecked")
    Callable<Object> query = () -> g.V(id)
      .fold()
      .coalesce(
        unfold(),
        addV("Person").property(T.id, id))
      .id().next();

    Status<Object> status = executor.execute(query);

    return status.getResult().toString();
  }

  private Cluster createCluster() {
    Cluster.Builder builder = Cluster.build()

.addContactPoint(System.getenv("NEPTUNE_ENDPOINT"))

.port(Integer.parseInt(System.getenv("NEPTUNE_PORT")))
                                    .enableSsl(true)
                                    .minConnectionPoolSize(1)
                                    .maxConnectionPoolSize(1)
                                    .serializer(Serializers.GRAPHBINARY_V1D0)
                                    .reconnectInterval(2000);

  if (Boolean.parseBoolean(getOptionalEnv("USE_IAM", "true"))) {
    // For versions of TinkerPop 3.4.11 or higher:
    builder.handshakeInterceptor( r ->
      {
        NeptuneNettyHttpSigV4Signer sigV4Signer = new
NeptuneNettyHttpSigV4Signer(region, new DefaultAWSCredentialsProviderChain());
```

```
        sigV4Signer.signRequest(r);
        return r;
      }
    )

    // Versions of TinkerPop prior to 3.4.11 should use the following approach.
    // Be sure to adjust the  imports to include:
    //   import org.apache.tinkerpop.gremlin.driver.SigV4WebSocketChannelizer;
    //   builder = builder.channelizer(SigV4WebSocketChannelizer.class);

    return builder.create();
  }

  private RetryConfig createRetryConfig() {
    return new RetryConfigBuilder().retryOnCustomExceptionLogic(retryLogic())
                                   .withDelayBetweenTries(1000, ChronoUnit.MILLIS)
                                   .withMaxNumberOfTries(5)
                                   .withFixedBackoff()
                                   .build();
  }

  private Function<Exception, Boolean> retryLogic() {
    return e -> {
      StringWriter stringWriter = new StringWriter();
      e.printStackTrace(new PrintWriter(stringWriter));
      String message = stringWriter.toString();

      // Check for connection issues
      if ( message.contains("Timed out while waiting for an available host") ||
           message.contains("Timed-out waiting for connection on Host") ||
           message.contains("Connection to server is no longer active") ||
           message.contains("Connection reset by peer") ||
           message.contains("SSLEngine closed already") ||
           message.contains("Pool is shutdown") ||
           message.contains("ExtendedClosedChannelException") ||
           message.contains("Broken pipe")) {
        return true;
      }

      // Concurrent writes can sometimes trigger a ConcurrentModificationException.
      // In these circumstances you may want to backoff and retry.
      if (message.contains("ConcurrentModificationException")) {
          return true;
      }
```

```
      // If the primary fails over to a new instance, existing connections to the old
 primary will
      // throw a ReadOnlyViolationException. You may want to back and retry.
      if (message.contains("ReadOnlyViolationException")) {
          return true;
      }

      return false;
    };
  }

  private String getOptionalEnv(String name, String defaultValue) {
    String value = System.getenv(name);
    if (value != null && value.length() > 0) {
      return value;
    } else {
      return defaultValue;
    }
  }
}
```

Se você quiser incluir a lógica de reconexão na função, consulte Exemplo de reconexão Java.

Exemplo de função do Lambda em JavaScript para Amazon Neptune

Observações sobre este exemplo

- O driver JavaScript não mantém um grupo de conexões. Ele sempre abre uma única conexão.
- O exemplo de função usa os utilitários de assinatura Sigv4 do gremlin-aws-sigv4 para assinar solicitações em um banco de dados habilitado para autenticação do IAM.
- Ele usa a função retry( ) do módulo utilitário assíncrono de código aberto para lidar com tentativas de recuo e repetição.
- As etapas do terminal do Gremlin exibe uma `promise` em JavaScript (consulte a documentação do TinkerPop). Para `next()`, isso é uma tupla `{value, done}`.
- Os erros de conexão são gerados no manipulador e tratados com alguma lógica de recuo e repetição, de acordo com as recomendações descritas aqui, com uma exceção. Há um tipo de problema de conexão que o driver não trata como uma exceção e que, portanto, não pode ser resolvido por essa lógica de recuo e repetição.

O problema é que, se uma conexão for fechada depois que um driver enviar uma solicitação, mas antes que o driver receba uma resposta, parecerá que a consulta está concluída, mas gera um valor nulo. No que diz respeito ao cliente da função do Lambda, a função parece ter sido concluída com êxito, mas com uma resposta vazia.

O impacto desse problema depende de como a aplicação trata uma resposta vazia. Algumas aplicações podem tratar uma resposta vazia de uma solicitação de leitura como um erro, mas outras podem tratá-la erroneamente como um resultado vazio.

Solicitações de gravação que encontrem esse problema de conexão também exibirão uma resposta vazia. Uma invocação bem-sucedida com uma resposta vazia indica êxito ou falha? Se o cliente que estiver invocando uma função de gravação simplesmente tratar a invocação bem-sucedida da função como se a gravação no banco de dados tivesse sido confirmada em vez de verificar o corpo da resposta, poderá parecer que o sistema perdeu dados.

Esse problema é causado pela forma como o driver trata os eventos emitidos pelo soquete subjacente. Quando o soquete de rede subjacente é fechado com um erro `ECONNRESET`, o WebSocket usado pelo driver é fechado e emite um evento `'ws close'`. No entanto, não há nada no driver que lide com esse evento de uma forma que possa ser usada para gerar uma exceção. Como resultado, a consulta simplesmente desaparece.

Para contornar esse problema, o exemplo da função do Lambda aqui adiciona um manipulador de eventos `'ws close'` que lança uma exceção ao driver ao criar uma conexão remota. No entanto, essa exceção não é gerada ao longo do caminho de solicitação-resposta da consulta do Gremlin e, portanto, não pode ser usada para acionar nenhuma lógica de recuo e repetição dentro da própria função do Lambda. Em vez disso, a exceção lançada pelo manipulador de eventos `'ws close'` gera uma exceção não tratada que faz com que a invocação do Lambda falhe. Isso permite ao cliente que invoca a função manipular o erro e repetir a invocação do Lambda, se apropriado.

Recomendamos que você implemente a lógica de recuo e repetição na própria função do Lambda para proteger os clientes contra problemas de conexão intermitentes. No entanto, a solução alternativa para o problema acima exige que o cliente implemente a lógica de repetição também para lidar com falhas resultantes desse problema de conexão específico.

Código Javascript

```
const gremlin = require('gremlin');
const async = require('async');
const {getUrlAndHeaders} = require('gremlin-aws-sigv4/lib/utils');

const traversal = gremlin.process.AnonymousTraversalSource.traversal;
const DriverRemoteConnection = gremlin.driver.DriverRemoteConnection;
const t = gremlin.process.t;
const __ = gremlin.process.statics;

let conn = null;
let g = null;

async function query(context) {

  const id = context.id;

  return g.V(id)
    .fold()
    .coalesce(
      __.unfold(),
      __.addV('User').property(t.id, id)
    )
    .id().next();
}

async function doQuery() {
  const id = Math.floor(Math.random() * 10000).toString();

  let result = await query({id: id});
  return result['value'];
}

exports.handler = async (event, context) => {

  const getConnectionDetails = () => {
    if (process.env['USE_IAM'] == 'true'){
       return getUrlAndHeaders(
         process.env['NEPTUNE_ENDPOINT'],
         process.env['NEPTUNE_PORT'],
         {},
         '/gremlin',
```

```
        'wss');
    } else {
      const database_url = 'wss://' + process.env['NEPTUNE_ENDPOINT'] + ':' +
process.env['NEPTUNE_PORT'] + '/gremlin';
      return { url: database_url, headers: {}};
    }
 };


 const createRemoteConnection = () => {
   const { url, headers } = getConnectionDetails();

   const c = new DriverRemoteConnection(
     url,
     {
       mimeType: 'application/vnd.gremlin-v2.0+json',
       headers: headers
     });

    c._client._connection.on('close', (code, message) => {
        console.info(`close - ${code} ${message}`);
        if (code == 1006){
          console.error('Connection closed prematurely');
          throw new Error('Connection closed prematurely');
        }
      });

    return c;
 };

 const createGraphTraversalSource = (conn) => {
   return traversal().withRemote(conn);
 };

 if (conn == null){
   console.info("Initializing connection")
   conn = createRemoteConnection();
   g = createGraphTraversalSource(conn);
 }

 return async.retry(
   {
     times: 5,
     interval: 1000,
```

```
      errorFilter: function (err) {

        // Add filters here to determine whether error can be retried
        console.warn('Determining whether retriable error: ' + err.message);

        // Check for connection issues
        if (err.message.startsWith('WebSocket is not open')){
          console.warn('Reopening connection');
          conn.close();
          conn = createRemoteConnection();
          g = createGraphTraversalSource(conn);
          return true;
        }

        // Check for ConcurrentModificationException
        if (err.message.includes('ConcurrentModificationException')){
          console.warn('Retrying query because of ConcurrentModificationException');
          return true;
        }

        // Check for ReadOnlyViolationException
        if (err.message.includes('ReadOnlyViolationException')){
          console.warn('Retrying query because of ReadOnlyViolationException');
          return true;
        }

        return false;
      }

    },
    doQuery);
};
```

Exemplo de função do Lambda em Python para Amazon Neptune

Veja algumas considerações sobre o seguinte exemplo de função AWS Lambda em Python:

- Ele usa o módulo de recuo.
- Ele define `pool_size=1` para evitar a criação de um grupo de conexões desnecessário.
- Ele define `message_serializer=serializer.GraphSONSerializersV2d0()`.

```
import os, sys, backoff, math
from random import randint
from gremlin_python import statics
from gremlin_python.driver.driver_remote_connection import DriverRemoteConnection
from gremlin_python.driver.protocol import GremlinServerError
from gremlin_python.driver import serializer
from gremlin_python.process.anonymous_traversal import traversal
from gremlin_python.process.graph_traversal import __
from gremlin_python.process.strategies import *
from gremlin_python.process.traversal import T
from aiohttp.client_exceptions import ClientConnectorError
from botocore.auth import SigV4Auth
from botocore.awsrequest import AWSRequest
from botocore.credentials import ReadOnlyCredentials
from types import SimpleNamespace

import logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)


reconnectable_err_msgs = [
    'ReadOnlyViolationException',
    'Server disconnected',
    'Connection refused',
    'Connection was already closed',
    'Connection was closed by server',
    'Failed to connect to server: HTTP Error code 403 - Forbidden'
]

retriable_err_msgs = ['ConcurrentModificationException'] + reconnectable_err_msgs

network_errors = [OSError, ClientConnectorError]

retriable_errors = [GremlinServerError, RuntimeError, Exception] + network_errors

def prepare_iamdb_request(database_url):

    service = 'neptune-db'
    method = 'GET'

    access_key = os.environ['AWS_ACCESS_KEY_ID']
    secret_key = os.environ['AWS_SECRET_ACCESS_KEY']
```

```
    region = os.environ['AWS_REGION']
    session_token = os.environ['AWS_SESSION_TOKEN']

    creds = SimpleNamespace(
        access_key=access_key, secret_key=secret_key, token=session_token,
 region=region,
    )

    request = AWSRequest(method=method, url=database_url, data=None)
    SigV4Auth(creds, service, region).add_auth(request)

    return (database_url, request.headers.items())

def is_retriable_error(e):

    is_retriable = False
    err_msg = str(e)

    if isinstance(e, tuple(network_errors)):
        is_retriable = True
    else:
        is_retriable = any(retriable_err_msg in err_msg for retriable_err_msg in
 retriable_err_msgs)

    logger.error('error: [{}] {}'.format(type(e), err_msg))
    logger.info('is_retriable: {}'.format(is_retriable))

    return is_retriable

def is_non_retriable_error(e):
    return not is_retriable_error(e)

def reset_connection_if_connection_issue(params):

    is_reconnectable = False

    e = sys.exc_info()[1]
    err_msg = str(e)

    if isinstance(e, tuple(network_errors)):
        is_reconnectable = True
    else:
        is_reconnectable = any(reconnectable_err_msg in err_msg for
 reconnectable_err_msg in reconnectable_err_msgs)
```

```
    logger.info('is_reconnectable: {}'.format(is_reconnectable))

    if is_reconnectable:
        global conn
        global g
        conn.close()
        conn = create_remote_connection()
        g = create_graph_traversal_source(conn)

@backoff.on_exception(backoff.constant,
    tuple(retriable_errors),
    max_tries=5,
    jitter=None,
    giveup=is_non_retriable_error,
    on_backoff=reset_connection_if_connection_issue,
    interval=1)
def query(**kwargs):

    id = kwargs['id']

    return (g.V(id)
        .fold()
        .coalesce(
            __.unfold(),
            __.addV('User').property(T.id, id)
        )
        .id().next())

def doQuery(event):
    return query(id=str(randint(0, 10000)))

def lambda_handler(event, context):
    result = doQuery(event)
    logger.info('result - {}'.format(result))
    return result

def create_graph_traversal_source(conn):
    return traversal().withRemote(conn)

def create_remote_connection():
    logger.info('Creating remote connection')

    (database_url, headers) = connection_info()
```

```
    return DriverRemoteConnection(
        database_url,
        'g',
        pool_size=1,
        message_serializer=serializer.GraphSONSerializersV2d0(),
        headers=headers)

def connection_info():

    database_url = 'wss://{}:{}/gremlin'.format(os.environ['neptuneEndpoint'],
 os.environ['neptunePort'])

    if 'USE_IAM' in os.environ and os.environ['USE_IAM'] == 'true':
        return prepare_iamdb_request(database_url)
    else:
        return (database_url, {})

conn = create_remote_connection()
g = create_graph_traversal_source(conn)
```

Veja exemplos de resultados, mostrando períodos alternados de carga pesada e leve:


GremlinWebSocketOpenConnections
Count
261
131
0
06:10 06:15 06:20 06:25 06:30 06:35 06:40 06:45 06:50 06:55 07:00 07:05
neptunedbinstance-l22qsahgvgvl
replica-python-lambdas-neptunebasestack-5rzhq9jvp2ri
GremlinRequestsPreSecond
Count/Second
861
430
0
06:10 06:15 06:20 06:25 06:30 06:35 06:40 06:45 06:50 06:55 07:00 07:05
neptunedbinstance-l22qsahgvgvl
replica-python-lambdas-neptunebasestack-5rzhq9jvp2ri
Lambda-ConcurrentExecutions
Count
86
43.5
1
06:10 06:15 06:20 06:25 06:30 06:35 06:40 06:45 06:50 06:55 07:00 07:05
social-b6501310_add-follows-relationship social-b6501310_add-follows-relationship
social-b6501310_create-user social-b6501310_create-user
social-b6501310_get-recent-follows social-b6501310_get-recent-follows
social-b6501310_get-user-details social-b6501310_get-user-details
Lambda-P99-Latency
Milliseconds
2.10k
1.05k
5.46
06:05 06:10 06:15 06:20 06:25 06:30 06:35 06:40 06:45 06:50 06:55 07:00 07:05
social-b6501310_add-follows-relationship
social-b6501310_create-user
social-b6501310_get-recent-follows
social-b6501310_get-user-details
social-b6501310_recommend-new-follows
Workload
Various units
Count
2.17k
1.09k
9e-3
147k
73.7k
2
06:10 06:15 06:20 06:25 06:30 06:35 06:40 06:45 06:50 06:55 07:00 07:05
P99Latency
AverageThroughput
Succeeded
Errors
Count
1
0.5
0
06:10 06:15 06:20 06:25 06:30 06:35 06:40 06:45 06:50 06:55 07:00 07:05
social-b6501310_add-follows-relationship
social-b6501310_create-user
social-b6501310_get-recent-follows
social-b6501310_get-user-details
social-b6501310_recommend-new-follows

Amazon Neptune ML para machine learning em grafos

Muitas vezes, há informações valiosas em grandes conjuntos de dados conectados que podem ser difíceis de extrair usando consultas baseadas apenas na intuição humana. As técnicas de machine learning (ML) podem ajudar a encontrar correlações ocultas em grafos com bilhões de relacionamentos. Essas correlações podem ser úteis para recomendar produtos, prever a capacidade de crédito, identificar fraudes e muitas outras coisas.

O atributo Neptune ML possibilita criar e treinar modelos úteis de machine learning úteis em grafos grandes em horas e não em semanas. Para isso, o Neptune ML usa a tecnologia de rede neural de grafos (GNN) desenvolvida pelo Amazon SageMaker e pela Deep Graph Library (DGL) (que é de código aberto). As redes neurais de grafos são um campo emergente em inteligência artificial (consulte, por exemplo, A Comprehensive Survey on Graph Neural Networks). Para ver um tutorial prático sobre como usar GNNs com a DGL, consulte Learning graph neural networks with Deep Graph Library.

> **Note**
>
> Os vértices do grafo são identificados nos modelos do Neptune ML como “nós”. Por exemplo, a classificação de vértices usa um modelo de machine learning de classificação de nós e a regressão de vértices usa um modelo de regressão de nós.

O que o Neptune ML pode fazer

O Neptune é compatível tanto com a inferência transdutiva, que gera previsões que foram pré-calculadas no momento do treinamento, com base nos dados de grafos da época, quanto a inferência indutiva, que se aplica ao processamento de dados e à avaliação de modelos em tempo real, com base nos dados atuais. Consulte A diferença entre inferência indutiva e transdutiva.

O Neptune ML pode treinar modelos de machine learning para oferecer compatibilidade com cinco categorias diferentes de inferência:

Tipos de tarefa de inferência atualmente compatíveis com o Neptune ML

- Classificação de nós: prevê o atributo categórico de uma propriedade de vértice.

Por exemplo, considerando-se o filme Um Sonho de Liberdade, o Neptune ML pode prever a propriedade `genre` como `story` partir de um conjunto candidato de `[story, crime, action, fantasy, drama, family, ...]`.

Há dois tipos de tarefas de classificação de nós:

- Classificação de classe única: nesse tipo de tarefa, cada nó tem apenas um atributo de destino. Por exemplo, a propriedade, `Place_of_birth` de `Alan Turing` tem o valor UK.
- Classificação de várias classes: nesse tipo de tarefa, cada nó pode ter mais de um atributo de destino. Por exemplo, a propriedade `genre` do filme O Poderoso Chefão tem os valores `crime` e `story`.

- Regressão de nós: prevê uma propriedade numérica de um vértice.

  Por exemplo, considerando-se o filme Vingadores: Ultimato, o Neptune ML pode prever que a propriedade `popularity` tem um valor de `5.0`.

- Classificação de bordas: prevê o atributo categórico de uma propriedade de borda.

  Há dois tipos de tarefas de classificação de bordas:

  - Classificação de classe única: nesse tipo de tarefa, cada borda tem apenas um atributo de destino. Por exemplo, uma borda de avaliação entre um usuário e um filme pode ter a propriedade `liked`, com um valor de “Sim” ou “Não”.
  - Classificação de várias classes: nesse tipo de tarefa, cada borda pode ter mais de um atributo de destino. Por exemplo, uma avaliação entre um usuário e um filme pode ter vários valores para a tag de propriedade, como “Engraçado”, “Emocionante”, “Assustador”, etc.

- Regressão de bordas: prevê uma propriedade numérica de uma borda.

  Por exemplo, uma borda de classificação entre um usuário e um filme pode ter a propriedade numérica `score` para a qual o Neptune ML pode prever um valor considerando-se a um usuário e um filme.

- Previsão de links: prevê os nós de destino mais prováveis para um nó de origem e uma borda de saída específicos, ou os nós de origem mais prováveis para um nó de destino e uma borda de entrada específicos.

  Por exemplo, com um grafo de conhecimento sobre medicamentos e doenças, fornecidos `Aspirin` como o nó de origem e `treats` como a borda de saída, o Neptune ML pode prever os nós de destino mais relevantes como `heart disease`, `fever`, etc.

Ou, com o gráfico de conhecimento da Wikimedia, fornecidos `President-of` como borda ou relação e `United-States` como nó de destino, o Neptune ML pode prever os presidentes mais relevantes como `George Washington`, `Abraham Lincoln`, `Franklin D. Roosevelt`, etc.

> **Note**
>
> A classificação de nós e a classificação de bordas são compatíveis somente com valores de string. Isso significa que valores de propriedades numéricas, como `0` ou `1`, não são compatíveis, embora a string seja equivalente a `"0"` e `"1"`. Da mesma forma, os valores de propriedade booliana `true` e `false` não funcionam, mas `"true"` e `"false"` funcionam.

Com o Neptune ML, é possível usar modelos de machine learning que se enquadram em duas categorias gerais:

Tipos de modelo de machine learning no momento compatíveis com o Neptune ML

- Modelos de rede neural de grafos (GNN): incluem Relational Graph Convolutional Networks (R-GCNs). Os modelos de GNN funcionam para todos os três tipos de tarefa acima.
- Modelos Knowledge-Graph Embedding (KGE): incluem modelos `TransE`, `DistMult` e `RotatE`. Eles só funcionam para previsão de links.

Modelos definidos pelo usuário: o Neptune ML também permite que você forneça sua própria implementação de modelos personalizados para todos os tipos de tarefas listados acima. Você pode usar o kit de ferramentas do Neptune ML para desenvolver e testar a implementação de modelos personalizados baseados em python antes de usar a API de treinamento do Neptune ML com o modelo. Consulte Modelos personalizados no Neptune ML para obter detalhes sobre como estruturar e organizar a implementação para que ela seja compatível com a infraestrutura de treinamento do Neptune ML.

Configurar o Neptune ML

A maneira mais fácil de começar a usar o Neptune ML é usar o modelo de início rápido do AWS CloudFormation. Esse modelo instala todos os componentes necessários, incluindo um novo cluster de banco de dados do Neptune, todos os perfis necessários do IAM e um novo bloco de anotações de grafos do Neptune para facilitar o trabalho com o Neptune ML.

Você também pode instalar o Neptune ML manualmente, conforme explicado em Configurar o Neptune ML sem usar o modelo de início rápido do AWS CloudFormation.

Usar o modelo AWS CloudFormation do Neptune ML para começar rapidamente em um novo cluster de banco de dados

A maneira mais fácil de começar a usar o Neptune ML é usar o modelo de início rápido do AWS CloudFormation. Esse modelo instala todos os componentes necessários, incluindo um novo cluster de banco de dados do Neptune, todos os perfis necessários do IAM e um novo bloco de anotações de grafos do Neptune para facilitar o trabalho com o Neptune ML.

Como criar a pilha de início rápido do Neptune ML

1. Para iniciar a pilha do AWS CloudFormation no console do AWS CloudFormation, selecione um dos botões Iniciar pilha na seguinte tabela:

Região	Visualização	Visualizar no Designer	Executar
Leste dos EUA (Norte da Virgínia)	Visão	Visualizar no Designer	Launch Stack
Leste dos EUA (Ohio)	Visão	Visualizar no Designer	Launch Stack
Oeste dos EUA (N. da Califórnia)	Visão	Visualizar no Designer	Launch Stack
Oeste dos EUA (Oregon)	Visão	Visualizar no Designer	Launch Stack
Canadá (Central)	Visão	Visualizar no Designer	Launch Stack
América do Sul (São Paulo)	Visão	Visualizar no Designer	Launch Stack
Europa (Estocolmo)	Visão	Visualizar no Designer	Launch Stack
Europa (Irlanda)	Visão	Visualizar no Designer	Launch Stack

Região	Visualização	Visualizar no Designer	Executar
Europa (Londres)	Visão	Visualizar no Designer	
Europa (Paris)	Visão	Visualizar no Designer	
Europa (Frankfurt)	Visão	Visualizar no Designer	
Oriente Médio (Barém)	Visão	Visualizar no Designer	
Oriente Médio (Emirados Árabes Unidos)	Visão	Visualizar no Designer	
Israel (Tel Aviv)	Visão	Visualizar no Designer	
África (Cidade do Cabo)	Visão	Visualizar no Designer	
Ásia-Pacífico (Hong Kong)	Visão	Visualizar no Designer	
Ásia-Pacífico (Tóquio)	Visão	Visualizar no Designer	
Ásia-Pacífico (Seul)	Visão	Visualizar no Designer	
Ásia-Pacífico (Singapura)	Visão	Visualizar no Designer	
Ásia-Pacífico (Sydney)	Visão	Visualizar no Designer	

Região	Visualização	Visualizar no Designer	Executar
Ásia-Pacífico (Mumbai)	Visão	Visualizar no Designer	
China (Pequim)	Visão	Visualizar no Designer	
China (Ningxia)	Visão	Visualizar no Designer	
AWS GovCloud (Oeste dos EUA)	Visão	Visualizar no Designer	

2. Na página Select Template, escolha Next.
3. Na página Specify Details (Especificar detalhes), escolha Next (Próximo).
4. Na página Options (Opções), escolha Next (Avançar).
5. Na página Revisar, há duas caixas de seleção que você precisa marcar:

   - A primeira confirma que o AWS CloudFormation pode criar recursos do IAM com nomes personalizados.
   - A segunda confirma que o AWS CloudFormation pode exigir a capacidade `CAPABILITY_AUTO_EXPAND` para a nova pilha. O `CAPABILITY_AUTO_EXPAND` permite explicitamente que o AWS CloudFormation expanda macros automaticamente ao criar a pilha, sem revisão prévia.

     Os clientes geralmente criam um conjunto de alterações a partir de um modelo processado para que as alterações feitas pelos macros possam ser revisadas antes de criar a pilha. Para obter mais informações, consulte a API CreateStack do AWS CloudFormation.

   Em seguida, selecione Criar.

O modelo de início rápido cria e configura o seguinte:

- Um cluster de banco de dados do Neptune
- Os perfis necessários do IAM (e os anexa).

- O grupo de segurança necessário do Amazon EC2.
- Os endpoints da VPC do SageMaker necessários.
- Um grupo de parâmetros de cluster de banco de dados para Neptune ML.
- Os parâmetros necessários nesse grupo de parâmetros.
- Um bloco de anotações do SageMaker com exemplos de bloco de anotações preenchidos previamente para o Neptune ML. Observe que nem todos os tamanhos de instância estão disponíveis em todas as regiões, então você precisa ter certeza de que o tamanho de instância do bloco de anotações selecionado é compatível com sua região.
- O serviço Neptune-Export.

Quando a pilha de início rápido estiver pronta, acesse o bloco de anotações do SageMaker criado pelo modelo e confira os exemplos preenchidos previamente. Eles ajudarão você a baixar exemplos de conjunto de dados para usar na experimentação dos recursos do Neptune ML.

Eles também podem proporcionar uma grande economia de tempo ao usar o Neptune ML. Por exemplo, veja a magia de linha %neptune_ml e a magia de célula %%neptune_ml compatíveis com esses blocos de anotações.

Você também pode usar o seguinte comando AWS CLI para executar o modelo de início rápido AWS CloudFormation:

```
aws cloudformation create-stack \
  --stack-name neptune-ml-fullstack-$(date '+%Y-%m-%d-%H-%M') \
  --template-url https://aws-neptune-customer-samples.s3.amazonaws.com/v2/
cloudformation-templates/neptune-ml-nested-stack.json \
  --parameters ParameterKey=EnableIAMAuthOnExportAPI,ParameterValue=(true if you have
 IAM auth enabled, or false otherwise) \
               ParameterKey=Env,ParameterValue=test$(date '+%H%M')\
  --capabilities CAPABILITY_IAM \
  --region (the AWS region, like us-east-1) \
  --disable-rollback \
  --profile (optionally, a named CLI profile of yours)
```

Configurar o Neptune ML sem usar o modelo de início rápido do AWS CloudFormation

1. Começar com um cluster de banco de dados do Neptune funcional

Se você não usar o modelo de início rápido do AWS CloudFormation para configurar o Neptune ML, precisará de um cluster de banco de dados do Neptune existente com o qual trabalhar. Se quiser, poderá usar um existente, clonar um que você já esteja usando ou criar outro (consulte Criar um cluster de banco de dados).

2. Instalar o serviço Neptune-Export

Se você ainda não o fez, instale o serviço Neptune Export, conforme explicado em Usar o serviço Neptune-Export para exportar dados do Neptune.

Adicione uma regra de entrada ao grupo de segurança `NeptuneExportSecurityGroup` criado pela instalação, com as seguintes configurações:

- Digite: `Custom TCP`
- Protocolo: `TCP`
- Intervalo de portas: `80 - 443`
- Fonte: *`(ID do grupo de segurança do cluster de banco de dados do Neptune)`*

3. Criar um perfil do IAM **`NeptuneLoadFromS3`** personalizado

Se você ainda não o fez, crie um perfil do IAM `NeptuneLoadFromS3` personalizado, conforme explicado em Criar um perfil do IAM para acessar o Amazon S3.

Criar um perfil **`NeptuneSageMakerIAMRole`** personalizado

Use o console do IAM para criar um `NeptuneSageMakerIAMRole` personalizado, usando a seguinte política:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ec2:CreateNetworkInterface",
```

```
        "ec2:CreateNetworkInterfacePermission",
        "ec2:CreateVpcEndpoint",
        "ec2:DeleteNetworkInterface",
        "ec2:DeleteNetworkInterfacePermission",
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeRouteTables",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcEndpoints",
        "ec2:DescribeVpcs"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "ecr:GetAuthorizationToken",
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage",
        "ecr:BatchCheckLayerAvailability"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/*"
      ],
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": [
            "sagemaker.amazonaws.com"
          ]
        }
      },
      "Effect": "Allow"
    },
    {
      "Action": [
        "kms:CreateGrant",
```

```
        "kms:Decrypt",
        "kms:GenerateDataKey*"
      ],
      "Resource": "arn:aws:kms:*:*:key/*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogGroups",
        "logs:DescribeLogStreams",
        "logs:GetLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:/aws/sagemaker/*"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "sagemaker:AddTags",
        "sagemaker:CreateEndpoint",
        "sagemaker:CreateEndpointConfig",
        "sagemaker:CreateHyperParameterTuningJob",
        "sagemaker:CreateModel",
        "sagemaker:CreateProcessingJob",
        "sagemaker:CreateTrainingJob",
        "sagemaker:CreateTransformJob",
        "sagemaker:DeleteEndpoint",
        "sagemaker:DeleteEndpointConfig",
        "sagemaker:DeleteModel",
        "sagemaker:DescribeEndpoint",
        "sagemaker:DescribeEndpointConfig",
        "sagemaker:DescribeHyperParameterTuningJob",
        "sagemaker:DescribeModel",
        "sagemaker:DescribeProcessingJob",
        "sagemaker:DescribeTrainingJob",
        "sagemaker:DescribeTransformJob",
        "sagemaker:InvokeEndpoint",
        "sagemaker:ListTags",
        "sagemaker:ListTrainingJobsForHyperParameterTuningJob",
        "sagemaker:StopHyperParameterTuningJob",
```

```
        "sagemaker:StopProcessingJob",
        "sagemaker:StopTrainingJob",
        "sagemaker:StopTransformJob",
        "sagemaker:UpdateEndpoint",
        "sagemaker:UpdateEndpointWeightsAndCapacities"
      ],
      "Resource": [
        "arn:aws:sagemaker:*:*:*"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "sagemaker:ListEndpointConfigs",
        "sagemaker:ListEndpoints",
        "sagemaker:ListHyperParameterTuningJobs",
        "sagemaker:ListModels",
        "sagemaker:ListProcessingJobs",
        "sagemaker:ListTrainingJobs",
        "sagemaker:ListTransformJobs"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject",
        "s3:AbortMultipartUpload",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

Ao criar esse perfil, edite o relacionamento de confiança da seguinte forma:

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "ec2.amazonaws.com",
          "rds.amazonaws.com",
          "sagemaker.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Por fim, copie o ARN atribuído a esse novo perfil `NeptuneSageMakerIAMRole`.

> **Important**
>
> - Garanta que as permissões do Amazon S3 no `NeptuneSageMakerIAMRole` correspondam às indicadas acima.
> - O ARN universal, `arn:aws:s3:::*`, é usado para o recurso do Amazon S3 na política acima. Se, por algum motivo, o ARN universal não puder ser usado, o `arn:aws:s3:::graphlytics*` e o ARN de qualquer outro recurso do Amazon S3 do cliente que os comandos do NeptuneML usarão deverá ser adicionado à seção de recursos.

Configurar o cluster de banco de dados do para habilitar o Neptune ML

Como configurar o cluster de banco de dados do Neptune ML

1. No console do Neptune, acesse Grupos de parâmetros e, depois, o grupo de parâmetros do cluster de banco de dados associado ao cluster de banco de dados que você usará. Defina o parâmetro `neptune_ml_iam_role` como o ARN atribuído ao perfil `NeptuneSageMakerIAMRole` que você acabou de criar.
2. Navegue até Bancos de dados e, depois, selecione o cluster de banco de dados que você usará para o Neptune ML. Selecione Ações e, depois, Gerenciar perfis do IAM.

3. Na página Gerenciar perfis do IAM, selecione Adicionar perfil e adicione o `NeptuneSageMakerIAMRole`. Depois, adicione o perfil `NeptuneLoadFromS3`.
4. Reinicialize a instância de gravador do cluster de banco de dados.

Criar dois endpoints do SageMaker na VPC do Neptune

Por fim, para permitir que o mecanismo do Neptune acesse as APIs de gerenciamento necessárias do SageMaker, é necessário criar dois endpoints do SageMaker na VPC do Neptune, conforme explicado em Criar dois endpoints para o SageMaker na VPC do Neptune.

Configurar manualmente um bloco de anotações Neptune para o Neptune ML

Os blocos de anotações SageMaker no Neptune vêm pré-carregados com uma série de exemplos de bloco de anotações para o Neptune ML. É possível visualizar esses exemplos no repositório do GitHub de blocos de anotações de grafos de código aberto.

É possível usar um dos blocos de anotações Neptune existentes ou, se quiser, criar um de sua preferência, seguindo as instruções em Usar a bancada de trabalho do Neptune para hospedar blocos de anotações Neptune.

Você também pode configurar um bloco de anotações Neptune padrão para uso com o Neptune ML seguindo estas etapas:

Modificar um bloco de anotações para o Neptune ML

1. Abra o console do Amazon SageMaker em https://console.aws.amazon.com/sagemaker/.
2. No painel de navegação à esquerda, selecione Bloco de anotações e, depois, Instâncias do bloco de anotações. Procure o nome do bloco de anotações Neptune que você gostaria de usar para o Neptune ML e selecione-o para acessar a página de detalhes.
3. Se a instância do bloco de anotações estiver em execução, selecione o botão Interromper, no canto superior direito da página de detalhes do bloco de anotações.
4. Nas Configurações da instância do bloco de anotações, em Configuração do ciclo de vida, selecione o link para abrir a página do ciclo de vida do bloco de anotações.
5. Selecione Editar no canto superior direito e, depois, Continuar.
6. Na guia Iniciar bloco de anotações, modifique o script para incluir comandos de exportação adicionais e preencher os campos do perfil do IAM do Neptune ML e o URI do serviço de exportação, algo parecido com o seguinte, dependendo do shell:

```
echo "export NEPTUNE_ML_ROLE_ARN=(your Neptune ML IAM role ARN)" >> ~/.bashrc
echo "export NEPTUNE_EXPORT_API_URI=(your export service URI)" >> ~/.bashrc
```

7. Selecione Atualizar.
8. Volte para a página da instância do bloco de anotações. Em Permissões e criptografia, há um campo para o ARN do perfil do IAM. Selecione o link nesse campo para acessar o perfil do IAM com o qual essa instância de bloco de anotações é executada.
9. Crie uma política em linha como esta:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "cloudwatch:PutMetricData"
      ],
      "Resource": "arn:aws:cloudwatch:[AWS_REGION]:[AWS_ACCOUNT_ID]:*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:DescribeLogStreams",
        "logs:PutLogEvents",
        "logs:GetLogEvents"
      ],
      "Resource": "arn:aws:logs:[AWS_REGION]:[AWS_ACCOUNT_ID]:*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "s3:Put*",
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": "arn:aws:s3:::*",
      "Effect": "Allow"
    },
    {
      "Action": "execute-api:Invoke",
```

```
      "Resource": "arn:aws:execute-api:[AWS_REGION]:[AWS_ACCOUNT_ID]:*/*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "sagemaker:CreateModel",
        "sagemaker:CreateEndpointConfig",
        "sagemaker:CreateEndpoint",
        "sagemaker:DescribeModel",
        "sagemaker:DescribeEndpointConfig",
        "sagemaker:DescribeEndpoint",
        "sagemaker:DeleteModel",
        "sagemaker:DeleteEndpointConfig",
        "sagemaker:DeleteEndpoint"
      ],
      "Resource": "arn:aws:sagemaker:[AWS_REGION]:[AWS_ACCOUNT_ID]:*/*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "[YOUR_NEPTUNE_ML_IAM_ROLE_ARN]"",
      "Effect": "Allow"
    }
  ]
}
```

10. Salve essa nova política e anexe-a ao perfil do IAM na etapa 8.
11. Selecione Iniciar no canto superior direito da instância do bloco de anotações do SageMaker para iniciar a instância do bloco de anotações.

Usar a AWS CLI para configurar o Neptune ML em um cluster de banco de dados

Além do modelo de início rápido do AWS CloudFormation e do AWS Management Console, também é possível configurar o Neptune ML usando a AWS CLI.

1. Criar um grupo de parâmetros de cluster de banco de dados para o novo cluster do Neptune ML

Os seguintes comandos AWS CLI criam um grupo de parâmetros do cluster de banco de dados e o configuram para funcionar com o Neptune ML:

Como criar e configurar um grupo de parâmetros de cluster de banco de dados para o Neptune ML

1. Crie um grupo de parâmetros de cluster de banco de dados:

```
aws neptune create-db-cluster-parameter-group \
  --db-cluster-parameter-group-name (name of the new DB cluster parameter group) \
  --db-parameter-group-family neptune1
  --description "(description of your machine learning project)" \
  --region (AWS region, such as us-east-1)
```

2. Crie um parâmetro de cluster de banco de dados `neptune_ml_iam_role` definido como o ARN do `SageMakerExcecutionIAMRole` para o cluster de banco de dados usar ao chamar o SageMaker para criar trabalhos e obter previsões de modelos de ML hospedados:

```
aws neptune modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name (name of the new DB cluster parameter group) \
  --parameters "ParameterName=neptune_ml_iam_role, \
                ParameterValue=ARN of the SageMakerExcecutionIAMRole, \
                Description=NeptuneMLRole, \
                ApplyMethod=pending-reboot" \
  --region (AWS region, such as us-east-1)
```

   Definir esse parâmetro permite ao Neptune acessar o SageMaker sem precisar transmitir o perfil com cada chamada.

   Para obter informações sobre como criar o `SageMakerExcecutionIAMRole`, consulte Criar um perfil `NeptuneSageMakerIAMRole` personalizado.

3. Por fim, use `describe-db-cluster-parameters` para conferir se todos os parâmetros no novo grupo de parâmetros do cluster de banco de dados estão definidos da forma desejada:

```
aws neptune describe-db-cluster-parameters \
  --db-cluster-parameter-group-name (name of the new DB cluster parameter group) \
  --region (AWS region, such as us-east-1)
```

Anexar o novo grupo de parâmetros de cluster de banco de dados ao cluster de banco de dados a ser usado com o Neptune ML

Agora é possível anexar o novo grupo de parâmetros do cluster de banco de dados que você acabou de criar a um cluster de banco de dados existente usando o seguinte comando:

```
aws neptune modify-db-cluster \
  --db-cluster-identifier (the name of your existing DB cluster) \
  --apply-immediately
  --db-cluster-parameter-group-name (name of your new DB cluster parameter group) \
  --region (AWS region, such as us-east-1)
```

Para tornar todos os parâmetros efetivos, você pode então reinicializar o cluster de banco de dados:

```
aws neptune reboot-db-instance
  --db-instance-identifier (name of the primary instance of your DB cluster) \
  --profile (name of your AWS profile to use) \
  --region (AWS region, such as us-east-1)
```

Ou, se estiver criando um cluster de banco de dados para usar com o Neptune ML, você poderá usar o seguinte comando para criar o cluster com o novo grupo de parâmetros anexado e, depois, criar uma instância principal (de gravador):

```
cluster-name=(the name of the new DB cluster)
aws neptune create-db-cluster
  --db-cluster-identifier ${cluster-name}
  --engine graphdb \
  --engine-version 1.0.4.1 \
  --db-cluster-parameter-group-name (name of your new DB cluster parameter group) \
  --db-subnet-group-name (name of the subnet to use) \
  --region (AWS region, such as us-east-1)
```

```
aws neptune create-db-instance
  --db-cluster-identifier ${cluster-name}
  --db-instance-identifier ${cluster-name}-i \
  --db-instance-class (the instance class to use, such as db.r5.xlarge)
  --engine graphdb \
  --region (AWS region, such as us-east-1)
```

Anexar o **`NeptuneSageMakerIAMRole`** ao cluster de banco de dados para que ele possa acessar os recursos do SageMaker e do Amazon S3.

Por fim, siga as instruções em Criar um perfil `NeptuneSageMakerIAMRole` personalizado para criar um perfil do IAM que permitirá ao cluster de banco de dados se comunicar com o SageMaker e o Amazon S3. Depois, use o seguinte comando para anexar o perfil `NeptuneSageMakerIAMRole` que você criou ao cluster de banco de dados:

```
aws neptune add-role-to-db-cluster
  --db-cluster-identifier ${cluster-name}
  --role-arn arn:aws:iam::(the ARN number of the role's ARN):role/NeptuneMLRole \
  --region (AWS region, such as us-east-1)
```

Criar dois endpoints para o SageMaker na VPC do Neptune

O Neptune ML precisa de dois endpoints do SageMaker na VPC do cluster de banco de dados do Neptune:

- `com.amazonaws.(AWS region, like us-east-1).sagemaker.runtime`
- `com.amazonaws.(AWS region, like us-east-1).sagemaker.api`

Se você não usou o modelo do AWS CloudFormation de início rápido, que os cria automaticamente, poderá usar os seguintes comandos da AWS CLI para criá-los:

Este cria o endpoint `sagemaker.runtime`:

```
create-vpc-endpoint
  --vpc-id (the ID of your Neptune DB cluster's VPC)
  --service-name com.amazonaws.(AWS region, like us-east-1).sagemaker.runtime
  --subnet-ids (the subnet ID or IDs that you want to use)
  --security-group-ids (the security group for the endpoint network interface, or omit
 to use the default)
```

```
  --private-dns-enabled
```

E este cria o endpoint `sagemaker.api`:

```
aws create-vpc-endpoint
  --vpc-id (the ID of your Neptune DB cluster's VPC)
  --service-name com.amazonaws.(AWS region, like us-east-1).sagemaker.api
  --subnet-ids (the subnet ID or IDs that you want to use)
  --security-group-ids (the security group for the endpoint network interface, or omit
 to use the default)
  --private-dns-enabled
```

Você também pode usar o console da VPC para criar esses endpoints. Consulte Secure prediction calls in Amazon SageMaker with AWS PrivateLink e Securing all Amazon SageMaker API calls with AWS PrivateLink.

Criar um parâmetro de endpoint de inferência do SageMaker no grupo de parâmetros de cluster de banco de dados

Para evitar a necessidade de especificar o endpoint de inferência do SageMaker do modelo que você está usando em cada consulta feita a ele, crie um parâmetro de cluster de banco de dados denominado `neptune_ml_endpoint` no grupo de parâmetros do cluster de banco de dados para o Neptune ML. Defina o parâmetro como o `id` do endpoint da instância em questão.

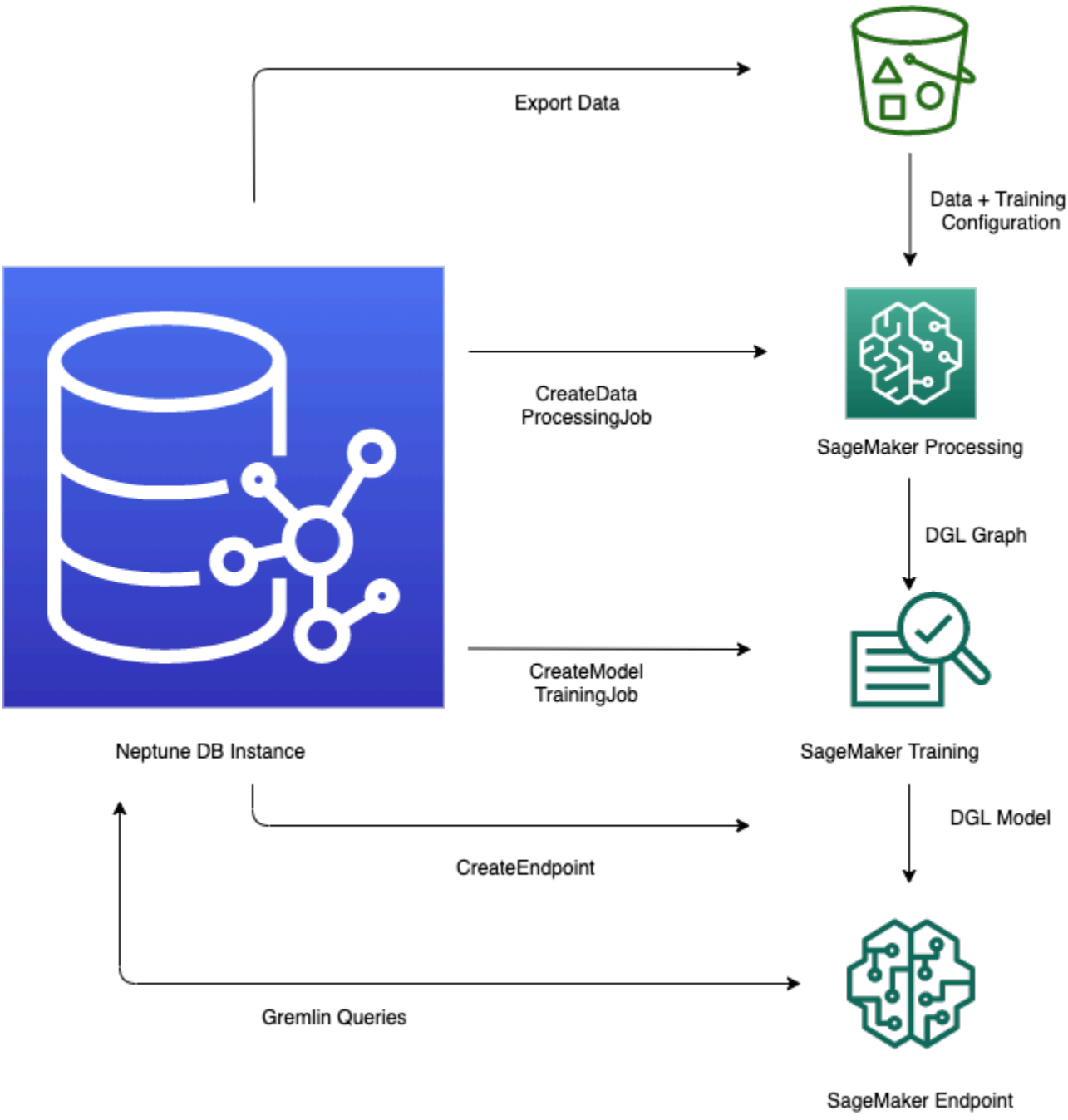
É possível usar o seguinte comando da AWS CLI para fazer isso:

```
aws neptune modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name neptune-ml-demo \
  --parameters "ParameterName=neptune_ml_endpoint, \
                ParameterValue=(the name of the SageMaker inference endpoint you want
 to query), \
                Description=NeptuneMLEndpoint, \
                ApplyMethod=pending-reboot" \
  --region (AWS region, such as us-east-1)
```

Visão geral de como usar o atributo Neptune ML

Iniciar o fluxo de trabalho para usar o Neptune ML

O uso do atributo Neptune ML no Amazon Neptune geralmente envolve estas cinco etapas para começar:



1. Exportação e configuração de dados: a etapa de exportação de dados usa o serviço Neptune-Export ou a ferramenta de linha de comando `neptune-export` para exportar dados do

Neptune para o Amazon Simple Storage Service (Amazon S3) em formato CSV. Um arquivo de configuração chamado `training-data-configuration.json` é gerado automaticamente ao mesmo tempo, o que especifica como os dados exportados podem ser carregados em um grafo treinável.

2. Pré-processamento de dados: nessa etapa, o conjunto de dados exportado é pré-processado usando técnicas padrão para prepará-lo para o treinamento de modelos. A normalização de atributos pode ser realizada para dados numéricos e os atributos de texto podem ser codificados usando `word2vec`. No final dessa etapa, um grafo DGL (biblioteca Deep Graph) é gerado a partir do conjunto de dados exportado para uso na etapa de treinamento de modelos.

   Essa etapa é implementada usando um trabalho de processamento do SageMaker na conta, e os dados resultantes são armazenados em um local do Amazon S3 especificado.

3. Treinamento de modelos: a etapa de treinamento de modelos treina o modelo de machine learning que será usado para previsões.

   O treinamento de modelos é realizado em duas etapas:

   - A primeira etapa usa um trabalho de processamento do SageMaker para gerar um conjunto de configurações da estratégia de treinamento de modelos que especifique o tipo de modelo e intervalos de hiperparâmetros que serão usados para o treinamento de modelos.
   - Depois, a segunda fase usa um trabalho de ajuste de modelos do SageMaker para testar diferentes configurações de hiperparâmetros e selecionar o trabalho de treinamento que produziu o modelo com melhor desempenho. O trabalho de ajuste executa um número pré-especificado de testes de trabalho de treinamento de modelos nos dados processados. No final dessa etapa, os parâmetros do modelo treinado do melhor trabalho de treinamento são usados para gerar artefatos de modelo para inferência.

4. Criação de um endpoint de inferência no Amazon SageMaker: o endpoint de inferência é uma instância de endpoint do SageMaker que é lançada com os artefatos do modelo produzidos pelo melhor trabalho de treinamento. Cada modelo é vinculado a um único endpoint. O endpoint pode aceitar solicitações recebidas do banco de dados de grafos e exibir as previsões do modelo para entradas nas solicitações. Depois de criar o endpoint, ele permanece ativo até que você o exclua.

5. Consulta ao modelo de machine learning usando o Gremlin: é possível usar extensões à linguagem de consulta Gremlin para consultar previsões por meio do endpoint de inferência.

> **Note**
>
> A bancada de trabalho do Neptune contém uma magia de linha e uma magia de célula que podem proporcionar uma grande economia de tempo no gerenciamento dessas etapas, ou seja:
>
> - %neptune_ml
> - %%neptune_ml

Fazendo previsões com base na evolução dos dados de grafos.

Com um grafo em constante mudança, convém criar previsões em lote periodicamente usando dados novos. Consultar previsões pré-calculadas (inferência transdutiva) pode ser significativamente mais rápido do que gerar novas previsões em tempo real com base nos dados mais recentes (inferência indutiva). As duas abordagens têm sua utilidade, dependendo da rapidez com que os dados mudam e dos requisitos de desempenho.

A diferença entre inferência indutiva e transdutiva

Ao realizar inferência transdutiva, o Neptune pesquisa e exibe previsões que foram pré-calculadas no momento do treinamento.

Ao realizar inferência indutiva, o Neptune cria o subgrafo relevante e busca as respectivas propriedades. O modelo DGL GNN então aplica o processamento de dados e a avaliação do modelo em tempo real.

Portanto, a inferência indutiva pode gerar previsões envolvendo nós e bordas que não estavam presentes no momento do treinamento e que refletem o estado atual do grafo. No entanto, isso ocorre à custa de uma maior latência.

Se o grafo for dinâmico, será conveniente usar a inferência indutiva para levar em conta os dados mais recentes, mas se o grafo for estático, a inferência transdutiva será mais rápida e eficiente.

A inferência indutiva é desabilitada por padrão. É possível habilitá-la para uma consulta usando o predicado Neptune#ml.inductiveInference do Gremlin na consulta da seguinte forma:

```
.with( "Neptune#ml.inductiveInference")
```

Fluxos transdutivos incrementais

Embora você atualize os artefatos do modelo simplesmente executando novamente as etapas de um a três (da Exportação e configuração de dados à Transformação de modelos), o Neptune ML aceita maneiras mais simples de atualizar as previsões de ML em lote usando novos dados. Uma delas é usar um fluxo de trabalho de modelo incremental e outra é usar o novo treinamento de modelos com uma inicialização a quente.

Fluxo de trabalho de modelo incremental

Nesse fluxo de trabalho, você atualiza as previsões de ML sem treinar o modelo de ML novamente.

> **Note**
>
> Você só poderá fazer isso quando os dados do grafo tiverem sido atualizados com novos nós e/ou bordas. No momento, ele não funcionará quando os nós forem removidos.

1. Exportação e configuração de dados: essa etapa é a mesma do fluxo de trabalho principal.
2. Pré-processamento incremental de dados: essa etapa é semelhante à etapa de pré-processamento de dados no fluxo de trabalho principal, mas usa a mesma configuração de processamento usada anteriormente, que corresponde a um modelo treinado específico.
3. Transformação de modelos: em vez de uma etapa de treinamento do modelo, essa etapa de transformação de modelos retira o modelo treinado do fluxo de trabalho principal e dos resultados da etapa de pré-processamento incremental de dados e gera novos artefatos do modelo para uso na inferência. A etapa de transformação de modelos inicia um trabalho de processamento do SageMaker para realizar o cálculo que gera os artefatos atualizados do modelo.
4. Atualização do endpoint de inferência do Amazon SageMaker: opcionalmente, se você tiver um endpoint de inferência existente, essa etapa atualizará o endpoint com os novos artefatos do modelo gerados pela etapa de transformação de modelos. Como alternativa, você também pode criar um endpoint de inferência com os novos artefatos do modelo.

Novo treinamento de modelos com uma inicialização a quente

Usando esse fluxo de trabalho, é possível treinar e implantar um novo modelo de ML para fazer previsões usando os dados de grafos incrementais, mas comece com um modelo existente gerado usando o fluxo de trabalho principal:

1. Exportação e configuração de dados: essa etapa é a mesma do fluxo de trabalho principal.
2. Pré-processamento incremental de dados: essa etapa é a mesma do fluxo de trabalho de inferência de modelo incremental. Os novos dados de grafos devem ser processados com o mesmo método de processamento usado anteriormente para o treinamento de modelos.
3. Treinamento de modelos com inicialização a quente: o treinamento de modelos é semelhante ao que acontece no fluxo de trabalho principal, mas é possível acelerar a pesquisa de hiperparâmetros do modelo aproveitando as informações da tarefa anterior de treinamento do modelo.
4. Atualização do endpoint de inferência do Amazon SageMaker: essa etapa é a mesma do fluxo de trabalho de inferência de modelos incrementais.

Fluxos de trabalho para modelos personalizados no Neptune ML

O Neptune ML permite implementar, treinar e implantar os próprios modelos personalizados para qualquer uma das tarefas compatíveis com o Neptune ML. O fluxo de trabalho para desenvolver e implantar um modelo personalizado é essencialmente o mesmo dos modelos integrados, com algumas diferenças, conforme explicado em Fluxo de trabalho de modelos personalizados.

Seleção de instâncias para os estágios do Neptune ML

Os diferentes estágios do processamento do Neptune ML usam diferentes instâncias do SageMaker. Aqui, abordamos como escolher o tipo de instância correto para cada estágio. É possível encontrar informações sobre os tipos e preços de instâncias do SageMaker em Preço do Amazon SageMaker.

Selecionar uma instância para processamento de dados

A etapa de processamento de dados do SageMaker requer uma instância de processamento que tenha memória e armazenamento em disco suficientes para os dados de entrada, intermediários e de saída. A quantidade específica de memória e armazenamento em disco necessária depende das características do grafo do Neptune ML e de os respectivos atributos exportados.

Por padrão, o Neptune ML escolhe a menor instância `ml.r5` cuja memória é dez vezes maior que o tamanho dos dados de grafos exportados no disco.

Selecionar uma instância para treinamento e transformação de modelos

Selecionar o tipo de instância certo para treinamento de modelos ou transformação de modelos depende do tipo de tarefa, do tamanho do grafo e dos requisitos de entrega. As instâncias de GPU oferecem o melhor desempenho. Geralmente, recomendamos instâncias seriais `p3` e `g4dn`. Também é possível usar instâncias `p2` ou `p4d`.

Por padrão, o Neptune ML escolhe a menor instância de GPU com mais memória do que o treinamento e a transformação de modelos exigem. É possível encontrar qual é essa seleção no arquivo `train_instance_recommendation.json`, no local de saída do processamento de dados do Amazon S3. Veja um exemplo do conteúdo de um arquivo `train_instance_recommendation.json`:

```
{
  "instance":     "(the recommended instance type for model training and transform)",
  "cpu_instance": "(the recommended instance type for base processing instance)",
  "disk_size":    "(the estimated disk space required)",
  "mem_size":     "(the estimated memory required)"
}
```

Selecionar uma instância para um endpoint de inferência

Selecionar o tipo de instância certo para um endpoint de inferência depende do tipo de tarefa, do tamanho do grafo e do seu orçamento. Por padrão, o Neptune ML escolhe a menor instância `ml.m5d` com mais memória do que o endpoint de inferência exige.

> **Note**
>
> Se forem necessários mais de 384 GB de memória, o Neptune ML usará uma instância `ml.r5d.24xlarge`.

É possível ver qual tipo de instância o Neptune ML recomenda no arquivo `infer_instance_recommendation.json` presente na localização do Amazon S3 que você está usando para treinamento de modelos. Veja uma amostra do conteúdo desse arquivo:

```
{
  "instance" :   "(the recommended instance type for an inference endpoint)",
  "disk_size" :  "(the estimated disk space required)",
  "mem_size" :   "(the estimated memory required)"
}
```

Usar a ferramenta **`neptune-export`** ou o serviço Neptune-Export para exportar dados do Neptune para o Neptune ML

O Neptune ML exige que você forneça dados de treinamento para a Deep Graph Library (DGL) criar e avaliar modelos.

É possível exportar dados do Neptune usando o serviço Neptune-Export ou o utilitário `neptune-export`. Tanto o serviço quanto a ferramenta de linha de comando publicam dados no Amazon Simple Storage Service (Amazon S3) em um formato CSV, criptografados usando criptografia no lado do servidor Amazon S3 (`SSE-S3`). Consulte Arquivos exportados pelo Neptune-Export e `neptune-export`.

Além disso, quando você configura uma exportação de dados de treinamento para o Neptune ML, o trabalho de exportação cria e publica um arquivo de configuração de treinamento de modelo criptografado junto com os dados exportados. Por padrão, esse arquivo é denominado `training-data-configuration.json`.

Exemplos de uso do serviço Neptune-Export para exportar dados de treinamento do Neptune ML

Essa solicitação exporta dados de treinamento do grafo de propriedades para uma tarefa de classificação de nós:

```
curl \
  (your NeptuneExportApiUri) \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "command": "export-pg",
        "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
        "params": {
          "endpoint": "(your Neptune endpoint DNS name)",
          "profile": "neptune_ml"
        },
        "additionalParams": {
          "neptune_ml": {
            "version": "v2.0",
            "targets": [
              {
                "node": "Movie",
```

```
                "property": "genre",
                "type": "classification"
              }
            ]
          }
        }
      }'
```

Essa solicitação exporta dados de treinamento do RDF para uma tarefa de classificação de nós:

```
curl \
  (your NeptuneExportApiUri) \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "command": "export-rdf",
        "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
        "params": {
          "endpoint": "(your Neptune endpoint DNS name)",
          "profile": "neptune_ml"
        },
        "additionalParams": {
          "neptune_ml": {
            "version": "v2.0",
            "targets": [
              {
                "node": "http://aws.amazon.com/neptune/csv2rdf/class/Movie",
                "predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/
genre",
                "type": "classification"
              }
            ]
          }
        }
      }'
```

Campos a serem definidos no objeto **`params`** ao exportar dados de treinamento

O objeto `params` em uma solicitação de exportação pode conter vários campos, conforme descrito na documentação de `params`. Veja os mais relevantes para exportar dados de treinamento de machine learning:

- **`endpoint`**: use `endpoint` para especificar um endpoint de uma instância do Neptune no cluster de banco de dados que o processo de exportação pode consultar para extrair dados.
- **`profile`**: o campo `profile` no objeto `params` deve ser definido como **`neptune-ml`**.

  Isso faz com que o processo de exportação formate os dados exportados adequadamente para o treinamento de modelos do Neptune ML, em formato CSV para dados de grafos de propriedades ou como N-Triples para dados do RDF. Isso também faz com que um arquivo `training-data-configuration.json` seja criado e gravado no mesmo local do Amazon S3 que os dados de treinamento exportados.
- **`cloneCluster`**: se definido como `true`, o processo de exportação clona o cluster de banco de dados, exporta do clone e, depois, exclui o clone ao terminar.
- **`useIamAuth`**: se o cluster de banco de dados tiver a autenticação do IAM habilitada, você deverá incluir esse campo definido como `true`.

O processo de exportação também fornece várias maneiras de filtrar os dados exportados (consulte estes exemplos).

Usar o objeto **`additionalParams`** para ajustar a exportação de informações de treinamento de modelos

O objeto `additionalParams` contém campos que você pode usar para especificar rótulos e atributos de classe de machine learning para fins de treinamento e orientar a criação de um arquivo de configuração de dados de treinamento.

O processo de exportação não pode inferir automaticamente quais propriedades do nó e da borda devem ser os rótulos das classes de machine learning para servir como exemplos para fins de treinamento. Ele também não pode inferir automaticamente a melhor codificação de atributos para propriedades numéricas, categóricas e de texto, portanto, você precisa fornecer dicas usando campos no objeto `additionalParams` para especificar esses itens ou substituir a codificação padrão.

Para dados de grafos de propriedades, a estrutura geral de `additionalParams` em uma solicitação de exportação pode ter a seguinte aparência:

```
{
  "command": "export-pg",
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
```

```
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "profile": "neptune_ml"
  },
  "additionalParams": {
      "neptune_ml": {
        "version": "v2.0",
        "targets": [ (an array of node and edge class label targets) ],
        "features": [ (an array of node feature hints) ]
    }
  }
}
```

Para dados do RDF, a estrutura geral pode ter a seguinte aparência:

```
{
  "command": "export-rdf",
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "profile": "neptune_ml"
  },
  "additionalParams": {
      "neptune_ml": {
        "version": "v2.0",
        "targets": [ (an array of node and edge class label targets) ]
    }
  }
}
```

Também é possível fornecer várias configurações de exportação usando o campo `jobs`:

```
{
  "command": "export-pg",
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "profile": "neptune_ml"
  },
  "additionalParams" : {
    "neptune_ml" : {
      "version": "v2.0",
      "jobs": [
```

```
        {
          "name" : "(training data configuration name)",
          "targets": [ (an array of node and edge class label targets) ],
          "features": [ (an array of node feature hints) ]
        },
        {
          "name" : "(another training data configuration name)",
          "targets": [ (an array of node and edge class label targets) ],
          "features": [ (an array of node feature hints) ]
        }
      ]
    }
  }
}
```

Elementos gerais no campo **`neptune_ml`** em **`additionalParams`**

O elemento **`version`** em **`neptune_ml`**

Especifica a versão da configuração dos dados de treinamento a ser gerada.

(Opcional), Tipo: string, Padrão: “v2.0”.

Se você incluir `version`, defina-o como `v2.0`.

O campo **`jobs`** no **`neptune_ml`**

Contém uma matriz de objetos de configuração de dados de treinamento, cada um definindo um trabalho de processamento de dados e contendo:

- **`name`**: o nome da configuração dos dados de treinamento a ser criada.

  Por exemplo, uma configuração de dados de treinamento com o nome “job-number-1" gera um arquivo de configuração de dados de treinamento chamado `job-number-1.json`.
- **`targets`**: uma matriz JSON de destinos de rótulos de classes de nós e bordas que representam os rótulos de classes de machine learning para fins de treinamento. Consulte O campo `targets` em um objeto `neptune_ml`.
- **`features`**: uma matriz JSON de atributos de propriedades de nós. Consulte O campo `features` no `neptune_ml`.

O campo **`targets`** em um objeto **`neptune_ml`**

O campo `targets` em uma configuração de exportação de dados de treinamento JSON contém uma matriz de objetos de destino que especificam uma tarefa de treinamento e os rótulos das classes de machine learning para treinar essa tarefa. O conteúdo dos objetos de destino varia dependendo se você está treinando com dados de grafos de propriedades ou dados do RDF.

Para tarefas de classificação e regressão de nós do grafo de propriedades, os objetos de destino na matriz podem ter a seguinte aparência:

```
{
  "node": "(node property-graph label)",
  "property": "(property name)",
  "type" : "(used to specify classification or regression)",
  "split_rate": [0.8,0.2,0.0],
  "separator": ","
}
```

Para tarefas de classificação de bordas, regressão ou previsão de links de grafos de propriedades, elas podem ter a seguinte aparência:

```
{
  "edge": "(edge property-graph label)",
  "property": "(property name)",
  "type" : "(used to specify classification, regression or link_prediction)",
  "split_rate": [0.8,0.2,0.0],
  "separator": ","
}
```

Para tarefas de classificação e regressão do RDF, os objetos de destino na matriz podem ter a seguinte aparência:

```
{
  "node": "(node type of an RDF node)",
  "predicate": "(predicate IRI)",
  "type" : "(used to specify classification or regression)",
  "split_rate": [0.8,0.2,0.0]
}
```

Para tarefas de previsão de links do RDF, os objetos de destino na matriz podem ter a seguinte aparência:

```
{
  "subject": "(source node type of an edge)",
  "predicate": "(relation type of an edge)",
  "object": "(destination node type of an edge)",
  "type" : "link_prediction",
  "split_rate": [0.8,0.2,0.0]
}
```

Os objetos de destino podem conter os seguintes campos:

Sumário

Campos em um objeto de destino do grafo de propriedades

O campo **node** (vértice) em um objeto de destino

O rótulo do grafo de propriedades de um nó de destino (vértice). Um objeto de destino deve conter um elemento `node` ou `edge`, mas não ambos.

O `node` pode assumir um único valor, como este:

```
"node": "Movie"
```

Ou, no caso de um vértice com vários rótulos, ele pode usar uma série de valores, como esta:

```
"node": ["Content", "Movie"]
```

O campo **`edge`** em um objeto de destino do grafo de propriedades

Especifica uma borda de destino pelos rótulos de nó inicial, o próprio rótulo e os rótulos de nó final. Um objeto de destino deve conter um elemento `edge` ou `node`, mas não ambos.

O valor de um campo `edge` é uma matriz JSON de três strings que representam os rótulos do grafo de propriedades do nó inicial, o rótulo do grafo de propriedades da própria borda e os rótulos do grafo de propriedades do nó final, desta forma:

```
"edge": ["Person_A", "knows", "Person_B"]
```

Se o nó inicial e/ou final tiver vários rótulos, coloque-os em uma matriz, como esta:

```
"edge": [ ["Admin", Person_A"], "knows", ["Admin", "Person_B"] ]
```

O campo **`property`** em um objeto de destino do grafo de propriedades

Especifica uma propriedade do vértice ou da borda de destino, desta forma:

```
"property" : "rating"
```

Esse campo é obrigatório, exceto quando a tarefa de destino é a previsão de links.

O campo **`type`** em um objeto de destino do grafo de propriedades

Indica o tipo de tarefa de destino a ser realizada no `node` ou na `edge`, desta forma:

```
"type" : "regression"
```

Os tipos de tarefa compatíveis com nós são:

- `classification`

- `regression`

Os tipos de tarefa compatíveis com bordas são:

- `classification`
- `regression`
- `link_prediction`

Este campo é obrigatório.

O campo **`split_rate`** em um objeto de destino do grafo de propriedades

(Opcional) Uma estimativa das proporções de nós ou bordas que as fases de treinamento, validação e teste usarão, respectivamente. Essas proporções são representadas por uma matriz JSON de três números entre zero e um que somam um:

```
"split_rate": [0.7, 0.1, 0.2]
```

Se você não fornecer o campo `split_rate` opcional, o valor estimado padrão será `[0.9, 0.1, 0.0]`.

O campo **`separator`** em um objeto de destino do grafo de propriedades

(Opcional) Usado com uma tarefa de classificação.

O campo `separator` especifica um caractere usado para dividir um valor de propriedade de destino em vários valores categóricos quando é usado para armazenar vários valores de categoria em uma string. Por exemplo:

```
"separator": "|"
```

A presença de um campo `separator` indica que a tarefa é de classificação com vários destinos.

Campos em um objeto de destino do RDF

O campo **`node`** em um objeto de destino do RDF

Define o tipo dos nós de destino. Usado com tarefas de classificação ou regressão de nós. O tipo de um nó no RDF é definido por:

```
node_id, <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>, node_type
```

Um node do RDF pode assumir um único valor, como este:

```
"node": "http://aws.amazon.com/neptune/csv2rdf/class/Movie"
```

O campo **`subject`** em um objeto de destino do RDF

Para tarefas de previsão de links, `subject` define o tipo de nó de origem das bordas de destino.

```
"subject": "http://aws.amazon.com/neptune/csv2rdf/class/Director"
```

> **Note**
>
> Para tarefas de previsão de links, `subject` deve ser usado junto com `predicate` e `object`. Se algum desses três não for fornecido, todas as bordas serão tratadas como o destino do treinamento.

O campo **`predicate`** em um objeto de destino do RDF

Para tarefas de classificação e regressão de nós, `predicate` define quais dados literais são usados como o atributo de um nó de destino.

```
"predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/genre"
```

> **Note**
>
> Se os nós de destino tiverem apenas um predicado que defina o atributo do nó de destino, o campo `predicate` poderá ser omitido.

Para tarefas de previsão de links, `predicate` define o tipo de relação das bordas de destino.

```
"predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/direct"
```

> **Note**
>
> Para tarefas de previsão de links, `predicate` deve ser usado junto com `subject` e `object`. Se algum desses três não for fornecido, todas as bordas serão tratadas como o destino do treinamento.

O campo **`object`** em um objeto de destino do RDF

Para tarefas de previsão de links, `object` define o tipo de nó de destino das bordas de destino:

```
"object": "http://aws.amazon.com/neptune/csv2rdf/class/Movie"
```

> **Note**
>
> Para tarefas de previsão de links, `object` deve ser usado junto com `subject` e `predicate`. Se algum desses três não for fornecido, todas as bordas serão tratadas como o destino do treinamento.

O campo **`type`** em um objeto de destino do RDF

Indica o tipo de tarefa de destino a ser realizada, desta forma:

```
"type" : "regression"
```

Os tipos de tarefa compatíveis com dados do RDF são:

- `link_prediction`
- `classification`
- `regression`

Este campo é obrigatório.

O campo **`split_rate`** em um objeto de destino do grafo de propriedades

(Opcional) Uma estimativa das proporções de nós ou bordas que as fases de treinamento, validação e teste usarão, respectivamente. Essas proporções são representadas por uma matriz JSON de três números entre zero e um que somam um:

```
"split_rate": [0.7, 0.1, 0.2]
```

Se você não fornecer o campo `split_rate` opcional, o valor estimado padrão será `[0.9, 0.1, 0.0]`.

O campo `features` no `neptune_ml`

Os valores das propriedades e os literais do RDF têm diferentes formatos e tipos de dados. Para obter um bom desempenho no machine learning, é essencial converter esses valores em codificações numéricas conhecidas como atributos.

O Neptune ML realiza extração e codificação de atributos como parte das etapas de exportação e processamento de dados, conforme descrito em Codificação de atributos no Neptune ML.

Para conjuntos de dados de grafos de propriedades, o processo de exportação infere automaticamente atributos `auto` para propriedades de string e propriedades numéricas que contêm vários valores. Para propriedades numéricas que contenham valores únicos, ele infere atributos `numerical`. Para propriedades de data, ele infere atributos `datetime`.

Se quiser substituir uma especificação de atributo inferida automaticamente ou adicionar uma especificação numérica de bucket, TF-IDF, FastText ou SBERT para uma propriedade, poderá controlar a codificação do atributo usando o campo atributos.

> **Note**
>
> É possível usar o campo `features` somente para controlar as especificações do atributo para dados de grafos de propriedades, não para dados do RDF.

Para texto de formato livre, o Neptune ML pode usar vários modelos diferentes para converter a sequência de tokens em um valor de propriedade de string em um vetor de valor real de tamanho fixo:

- `text_fasttext`: usa a codificação fastText. É a codificação recomendada para atributos que usam um e somente um dos cinco idiomas aceitos pela codificação fastText.
- `text_sbert`: usa os modelos de codificação Sentence BERT (SBERT). É a codificação recomendada para texto que não é compatível com `text_fasttext`.
- `text_word2vec`: usa algoritmos Word2Vec originalmente publicados pelo Google para codificar texto. O Word2Vec é compatível apenas com inglês.
- `text_tfidf`: usa um vetorizador de frequência de termo – frequência inversa do documento (TF-IDF) para codificar texto. A codificação TF-IDF é compatível com atributos estatísticos não aceitos por outras codificações.

O campo `features` contém uma matriz JSON de atributos de propriedade do nó. Os objetos na matriz podem conter os seguintes campos:

Sumário

O campo **node** em **features**

O campo node especifica um rótulo de grafo de propriedades de um vértice de atributo. Por exemplo:

```
  "node": "Person"
```

Se um vértice tiver vários rótulos, use uma matriz para contê-los. Por exemplo:

```
  "node": ["Admin", "Person"]
```

O campo **edge** no **features**

O campo `edge` especifica o tipo de uma borda de atributo. Um tipo de borda consiste em uma matriz que contém os rótulos do grafo de propriedades do vértice de origem, o rótulo do grafo de propriedades da borda e os rótulos do grafo de propriedades do vértice de destino. É necessário fornecer todos os três valores ao especificar um atributo de borda. Por exemplo:

```
"edge": ["User", "reviewed", "Movie"]
```

Se um vértice de origem ou destino de um tipo de borda tiver vários rótulos, use outra matriz para contê-los. Por exemplo:

```
"edge": [["Admin", "Person"]. "edited", "Post"]
```

O campo **property** no **features**

Use o parâmetro propriedade para especificar uma propriedade do vértice identificado pelo parâmetro `node`. Por exemplo:

```
"property" : "age"
```

Valores possíveis do campo **type** para atributos

O parâmetro `type` especifica o tipo de atributo que está sendo definido. Por exemplo:

```
"type": "bucket_numerical"
```

Valores possíveis do parâmetro **type**

- **"auto"**: especifica que o Neptune ML deve detectar automaticamente o tipo de propriedade e aplicar uma codificação de atributo adequada. Um atributo `auto` também pode ter um campo `separator` opcional.

  Consulte Codificação de atributos automáticos no Neptune ML.
- **"category"**: essa codificação de atributo representa um valor de propriedade como uma das várias categorias. Em outras palavras, o atributo pode assumir um ou mais valores distintos. Um atributo `category` também pode ter um campo `separator` opcional.

Consulte Atributos categóricos no Neptune ML.

- **`"numerical"`**: essa codificação de atributo representa valores de propriedade numérica como números em um intervalo contínuo em que “maior que” e “menor que” têm significado.

  Um atributo `numerical` também pode ter campos `norm`, `imputer` e `separator` opcionais.

  Consulte Atributos numéricos no Neptune ML.

- **`"bucket_numerical"`**: essa codificação de atributos divide os valores de propriedade numérica em um conjunto de buckets ou categorias.

  Por exemplo, é possível codificar a idade das pessoas em quatro buckets: crianças (de 0 a 20 anos), jovens adultos (de 20 a 40 anos), meia idade (de 40 a 60 anos) e idosos (de 60 anos em diante).

  Um atributo `bucket_numerical` requer um campo `range` e um `bucket_cnt` e também pode incluir um campo `imputer` e/ou `slide_window_size`.

  Consulte Atributos numéricos de bucket no Neptune ML.

- **`"datetime"`**: essa codificação de atributos representa um valor de propriedade de data e hora como uma matriz desses atributos categóricos: ano, mês, dia da semana e hora.

  Uma ou mais dessas quatro categorias podem ser eliminadas usando o parâmetro `datetime_parts`.

  Consulte Atributos de data e hora no Neptune ML.

- **`"text_fasttext"`**: essa codificação de atributos converte valores de propriedade que consistem em frases ou texto de formato livre em vetores numéricos usando modelos fastText. Ela é compatível com cinco idiomas, a saber, inglês (`en`), chinês (`zh`), hindi (`hi`), espanhol (`es`) e francês (`fr`). Para valores de propriedade de texto em qualquer um desses cinco idiomas, `text_fasttext` é a codificação recomendada. No entanto, ela não consegue lidar com casos em que a mesma frase contenha palavras em mais de um idioma.

  Para outros idiomas além dos compatíveis com a fastText, use a codificação `text_sbert`.

  Se você tiver muitas strings de texto de valor de propriedade maiores que, digamos, 120 tokens, use o campo `max_length` para limitar o número de tokens em cada string codificada por `"text_fasttext"`.

Consulte Codificação fastText de valores de propriedades de texto no Neptune ML.

- **`"text_sbert"`**: essa codificação converte valores de propriedade de texto em vetores numéricos usando modelos Sentence BERT (SBERT). O Neptune é compatível com dois métodos SBERT, ou seja, `text_sbert128`, que será o padrão se você apenas especificar `text_sbert` e `text_sbert512`. A diferença entre eles é o número máximo de tokens em uma propriedade de texto codificada. A codificação `text_sbert128` codifica apenas os primeiros 128 tokens, enquanto `text_sbert512` codifica até 512 tokens. Como resultado, o uso de `text_sbert512` pode exigir mais tempo de processamento do que `text_sbert128`. Os dois métodos são mais lentos do que `text_fasttext`.

  Os métodos `text_sbert*` são compatíveis com vários idiomas e podem codificar uma frase que contenha mais de um idioma.

  Consulte Codificação Sentence BERT (SBERT) de atributos de texto no Neptune ML.
- **`"text_word2vec"`**: essa codificação converte valores de propriedade de texto em vetores numéricos usando algoritmos Word2Vec. Ela aceita apenas inglês.

  Consulte Codificação Word2Vec de atributos de texto no Neptune ML.
- **`"text_tfidf"`**: essa codificação converte valores de propriedade de texto em vetores numéricos usando um vetorizador de frequência de termo – frequência inversa de documento (TF-IDF).

  Você define os parâmetros de uma codificação de atributos `text_tfidf` usando os campos `ngram_range`, `min_df` e `max_features`.

  Consulte Codificação TF-IDF de atributos de texto no Neptune ML.
- **`"none"`**: usar o tipo `none` faz com que nenhuma codificação de atributos ocorra. Em vez disso, os valores brutos das propriedades são analisados e salvos.

  Use `none` somente se você planeja realizar a própria codificação de atributos personalizados como parte do treinamento de modelos personalizados.

O campo **`norm`**

Esse campo é obrigatório para atributos numéricos. Ele especifica um método de normalização a ser usado em valores numéricos:

```
"norm": "min-max"
```

Os seguintes métodos de normalização são compatíveis:

- “mín-máx”: normalize cada valor subtraindo o valor mínimo e dividindo-o pela diferença entre o valor máximo e o mínimo.
- “padrão”: normalize cada valor dividindo-o pela soma de todos os valores.
- “nenhum”: não normalize os valores numéricos durante a codificação.

Consulte Atributos numéricos no Neptune ML.

O campo **`language`**

O campo de idioma especifica o idioma usado nos valores das propriedades de texto. O uso depende do método de codificação de texto:

- Para a codificação `text_fasttext`, esse campo é obrigatório e deve especificar um dos seguintes idiomas:
  - `en` (inglês)
  - `zh` (chinês)
  - `hi` (hindi)
  - `es` (espanhol)
  - `fr` (francês)
- Para a codificação `text_sbert`, esse campo não é usado, pois a codificação SBERT é multilíngue.
- Para a codificação `text_word2vec`, esse campo é opcional, pois `text_word2vec` só é compatível com inglês. Se presente, ele deve especificar o nome do modelo em inglês:

```
"language" : "en_core_web_lg"
```

- Para a codificação `text_tfidf`, esse campo não é usado.

O campo **`max_length`**

O campo `max_length` é opcional para atributos `text_fasttext`, no qual ele especifica o número máximo de tokens em um atributo de texto de entrada que será codificado. Texto de entrada maior do que `max_length` é truncado. Por exemplo, definir max_length como 128 indica que qualquer token após o 128º em uma sequência de texto será ignorado:

```
"max_length": 128
```

O campo **separator**

Esse campo é usado opcionalmente com os atributos `category`, `numerical` e `auto`. Ele especifica um caractere que pode ser usado para dividir o valor de uma propriedade em vários valores categóricos ou valores numéricos:

```
"separator": ";"
```

Use o campo `separator` somente quando a propriedade armazena vários valores delimitados em uma única string, como `"Actor;Director"` ou `"0.1;0.2"`.

Consulte Atributos categóricos, Atributos numéricos e Codificação automática.

O campo **range**

Esse campo é obrigatório para atributos `bucket_numerical`. Ele especifica o intervalo de valores numéricos que devem ser divididos em buckets, no formato `[`*`lower-bound`*`, `*`upper-bound`*`]`:

```
"range" : [20, 100]
```

Se o valor de uma propriedade for menor que o limite inferior, ele será atribuído ao primeiro bucket ou, se for maior que o limite superior, será atribuído ao último bucket.

Consulte Atributos numéricos de bucket no Neptune ML.

O campo **bucket_cnt**

Esse campo é obrigatório para atributos `bucket_numerical`. Ele especifica o número de buckets nos quais o intervalo numérico definido pelo parâmetro `range` deve ser dividido:

```
"bucket_cnt": 10
```

Consulte Atributos numéricos de bucket no Neptune ML.

O campo **slide_window_size**

Esse campo é usado opcionalmente com atributos `bucket_numerical` para atribuir valores a mais de um bucket:

```
"slide_window_size": 5
```

Uma janela de controle deslizante funciona da seguinte forma: o Neptune ML considera o tamanho da janela **s** e transforma cada valor numérico **v** de uma propriedade em um intervalo de `v - s/2` a `v + s/2` . O valor é então atribuído a cada bucket ao qual o intervalo se sobrepõe.

Consulte Atributos numéricos de bucket no Neptune ML.

O campo **imputer**

Esse campo é usado opcionalmente com atributos `numerical` e `bucket_numerical` para fornecer uma técnica de imputação a fim de preencher valores ausentes:

```
"imputer": "mean"
```

As técnicas de imputação compatíveis são:

- `"mean"`
- `"median"`
- `"most-frequent"`

Se você não incluir o parâmetro estimador, o pré-processamento de dados será interrompido e encerrado quando um valor ausente for encontrado.

Consulte Atributos numéricos no Neptune ML e Atributos numéricos de bucket no Neptune ML.

O campo **max_features**

Esse campo é usado opcionalmente pelos atributos `text_tfidf` para especificar o número máximo de termos a serem codificados:

```
"max_features": 100
```

Uma configuração de cem faz com que o vetorizador TF-IDF codifique somente os cem termos mais comuns. O valor padrão, se você não incluir `max_features` será cinco mil.

Consulte Codificação TF-IDF de atributos de texto no Neptune ML.

O campo **min_df**

Esse campo é usado opcionalmente pelos atributos `text_tfidf` para especificar a frequência mínima de documentos de termos a serem codificados:

```
"min_df": 5
```

Uma configuração de cinco indica que um termo deve aparecer em pelo menos cinco valores de propriedade diferentes para ser codificado.

O valor padrão se você não incluir o parâmetro `min_df` será `2`.

Consulte Codificação TF-IDF de atributos de texto no Neptune ML.

O campo **ngram_range**

Esse campo é usado opcionalmente pelos atributos `text_tfidf` para especificar quais sequências de tamanho de palavras ou tokens devem ser consideradas como possíveis termos individuais a serem codificados:

```
"ngram_range": [2, 4]
```

O valor `[2, 4]` especifica que sequências de dois, três e quatro palavras devem ser consideradas como possíveis termos individuais.

O padrão, se você não definir explicitamente `ngram_range` será `[1, 1]`, o que significa que somente palavras ou tokens individuais são considerados termos a serem codificados.

Consulte Codificação TF-IDF de atributos de texto no Neptune ML.

O campo **datetime_parts**

Esse campo é usado opcionalmente pelos atributos `datetime` para especificar quais partes do valor de data e hora devem ser codificadas categoricamente:

```
"datetime_parts": ["weekday", "hour"]
```

Se você não incluir `datetime_parts`, por padrão, o Neptune ML codificará as partes de ano, mês, dia da semana e hora do valor de data e hora. O valor `["weekday", "hour"]` indica que somente

os valores do dia da semana e da hora da data e hora devem ser codificados categoricamente no atributo.

Se uma das partes não tiver mais de um valor exclusivo no conjunto de treinamento, ela não será codificada.

Consulte Atributos de data e hora no Neptune ML.

Exemplos de uso de parâmetros em **`additionalParams`** para ajustar a configuração de treinamento de modelos

Sumário

Exemplos de grafo de propriedades usando **`additionalParams`**

Especificar uma taxa de divisão padrão para a configuração de treinamento de modelos

No exemplo a seguir, o parâmetro `split_rate` define a taxa de divisão padrão para o treinamento de modelos. Se nenhuma taxa de divisão padrão for especificada, o treinamento usará um valor de [0,9, 0,1, 0,0]. É possível substituir o valor padrão por destino especificando uma `split_rate` para cada destino.

No seguinte exemplo, o campo `default split_rate` indica que uma taxa de divisão de `[0.7,0.1,0.2]` deve ser usada, a menos que seja substituída por destino:”

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "split_rate": [0.7,0.1,0.2],
    "targets": [
      (...)
    ],
    "features": [
      (...)
    ]
  }
}
```

Especificar uma tarefa de classificação de nós para configuração de treinamento de modelos

Para indicar qual propriedade de nó contém exemplos rotulados para fins de treinamento, adicione um elemento de classificação de nós à matriz `targets`, usando `"type" : "classification"`. Adicione um campo `split_rate` se quiser substituir a taxa de divisão padrão.

No exemplo a seguir, o destino `node` indica que a propriedade `genre` de cada nó `Movie` deve ser tratada como um rótulo de classe de nó. O valor `split_rate` substitui a taxa de divisão padrão:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "node": "Movie",
        "property": "genre",
```

```
        "type": "classification",
        "split_rate": [0.7,0.1,0.2]
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Especificar uma tarefa de classificação de nós de várias classes para configuração de treinamento de modelos

Para indicar qual propriedade de nó contém vários exemplos rotulados para fins de treinamento, adicione um elemento de classificação de nós à matriz de destinos, usando `"type" : "classification"` e `separator` para especificar um caractere que pode ser usado para dividir o valor de uma propriedade de destino em vários valores categóricos. Adicione um campo `split_rate` se quiser substituir a taxa de divisão padrão.

No exemplo a seguir, o destino `node` indica que a propriedade `genre` de cada nó `Movie` deve ser tratada como um rótulo de classe de nó. O campo `separator` indica que cada propriedade de gênero contém vários valores separados por ponto e vírgula:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "node": "Movie",
        "property": "genre",
        "type": "classification",
        "separator": ";"
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Especificar uma tarefa de regressão de nós para configuração de treinamento de modelos

Para indicar qual propriedade de nó contém regressões rotuladas para fins de treinamento, adicione um elemento de regressão de nós à matriz de destinos, usando `"type" : "regression"`. Adicione um campo split_rate se quiser substituir a taxa de divisão padrão.

O seguinte destino `node` indica que a propriedade `rating` de cada nó `Movie` deve ser tratada como um rótulo de regressão de nós:

```
    "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "node": "Movie",
        "property": "rating",
        "type" : "regression",
        "split_rate": [0.7,0.1,0.2]
      }
    ],
    "features": [
      ...
    ]
  }
}
```

Especificar uma tarefa de classificação de bordas para configuração de treinamento de modelos

Para indicar qual propriedade de borda contém exemplos rotulados para fins de treinamento, adicione um elemento de borda à matriz `targets`, usando `"type" : "regression"`. Adicione um campo split_rate se quiser substituir a taxa de divisão padrão.

O seguinte destino `edge` indica que a propriedade `metAtLocation` de cada borda `knows` deve ser tratada como um rótulo de classe de borda:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "edge": ["Person", "knows", "Person"],
```

```
        "property": "metAtLocation",
        "type": "classification"
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Especificar uma tarefa de classificação de bordas de várias classes para configuração de treinamento de modelos

Para indicar qual propriedade de nó contém vários exemplos rotulados para fins de treinamento, adicione um elemento de borda à matriz `targets` usando um campo `"type" : "classification"` e um `separator` para especificar um caractere usado para dividir o valor de uma propriedade de destino em vários valores categóricos. Adicione um campo `split_rate` se quiser substituir a taxa de divisão padrão.

O destino `edge` a seguir indica que a propriedade `sentiment` de cada borda `repliedTo` deve ser tratada como um rótulo de classe de borda. O campo separador indica que cada propriedade de sentimento contém vários valores separados por vírgula:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "edge": ["Person", "repliedTo", "Message"],
        "property": "sentiment",
        "type": "classification",
        "separator": ","
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Especificar uma regressão de bordas para configuração de treinamento de modelos

Para indicar qual propriedade de borda contém exemplos de regressão rotulados para fins de treinamento, adicione um elemento `edge` à matriz `targets`, usando `"type" : "regression"`. Adicione um campo `split_rate` se quiser substituir a taxa de divisão padrão.

O seguinte destino `edge` indica que a propriedade `rating` de cada borda `reviewed` deve ser tratada como uma regressão de bordas:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "edge": ["Person", "reviewed", "Movie"],
        "property": "rating",
        "type" : "regression"
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Especificar uma tarefa de previsão de links para configuração de treinamento de modelos

Para indicar quais bordas devem ser usadas para fins de treinamento de previsão de links, adicione um elemento de borda à matriz de destinos usando `"type" : "link_prediction"`. Adicione um campo `split_rate` se quiser substituir a taxa de divisão padrão.

O seguinte destino `edge` indica que as bordas `cites` devem ser usadas para previsão de links:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "edge": ["Article", "cites", "Article"],
        "type" : "link_prediction"
      }
    ],
```

```
    "features": [
      (...)
    ]
  }
}
```

Especificar um atributo de bucket numérico

É possível especificar um atributo de dados numéricos para uma propriedade de nó adicionando `"type": "bucket_numerical"` à matriz `features`.

O seguinte atributo node indica que a propriedade `age` de cada nó `Person` deve ser tratada como um atributo de bucket numérico:

```
  "additionalParams": {
  "neptune_ml": {
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Person",
        "property": "age",
        "type": "bucket_numerical",
        "range": [1, 100],
        "bucket_cnt": 5,
        "slide_window_size": 3,
        "imputer": "median"
      }
    ]
  }
}
```

Especificar um atributo **`Word2Vec`**

É possível especificar um atributo `Word2Vec` para uma propriedade de nó adicionando `"type": "text_word2vec"` à matriz `features`.

O seguinte atributo node indica que a propriedade `description` de cada nó `Movie` deve ser tratada como um atributo `Word2Vec`:

```
"additionalParams": {
```

```
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Movie",
        "property": "description",
        "type": "text_word2vec",
        "language": "en_core_web_lg"
      }
    ]
  }
}
```

Especificar um atributo **`FastText`**

É possível especificar um atributo `FastText` para uma propriedade de nó adicionando `"type": "text_fasttext"` à matriz `features`. O campo `language` é obrigatório e deve especificar um dos seguintes códigos de idioma:

- `en` (inglês)
- `zh` (chinês)
- `hi` (hindi)
- `es` (espanhol)
- `fr` (francês)

Observe que a codificação `text_fasttext` não pode lidar com mais de um idioma por vez em um atributo.

O seguinte atributo node indica que a propriedade `description` de francês de cada nó `Movie` deve ser tratada como um atributo `FastText`:

```
"additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
```

```
    ],
    "features": [
      {
        "node": "Movie",
        "property": "description",
        "type": "text_fasttext",
        "language": "fr",
        "max_length": 1024
      }
    ]
  }
}
```

Especificar um atributo **`Sentence BERT`**

É possível especificar um atributo `Sentence BERT` para uma propriedade de nó adicionando `"type": "text_sbert"` à matriz `features`. Não é necessário especificar o idioma, pois o método codifica automaticamente os atributos de texto usando um modelo de idioma multilíngue.

O seguinte atributo node indica que a propriedade `description` de cada nó `Movie` deve ser tratada como um atributo `Sentence BERT`:

```
"additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Movie",
        "property": "description",
        "type": "text_sbert128",
      }
    ]
  }
}
```

Especificar um atributo **`TF-IDF`**

É possível especificar um atributo `TF-IDF` para uma propriedade de nó adicionando `"type": "text_tfidf"` à matriz `features`.

O seguinte atributo node indica que a propriedade `bio` de cada nó `Person` deve ser tratada como um atributo `TF-IDF`:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Movie",
        "property": "bio",
        "type": "text_tfidf",
        "ngram_range": [1, 2],
        "min_df": 5,
        "max_features": 1000
      }
    ]
  }
}
```

Especificar um atributo **`datetime`**

O processo de exportação infere automaticamente atributos `datetime` para propriedades de data. No entanto, se você quiser limitar `datetime_parts` usado de um atributo `datetime` ou substituir uma especificação de atributo para que uma propriedade que normalmente seria tratada como um atributo `auto` seja explicitamente tratada como um atributo `datetime`, você pode fazer isso adicionando `"type": "datetime"` a à matriz de atributos.

O seguinte atributo node indica que a propriedade `createdAt` de cada nó `Post` deve ser tratada como um atributo `datetime`:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Post",
```

```
        "property": "createdAt",
        "type": "datetime",
        "datetime_parts": ["month", "weekday", "hour"]
      }
    ]
  }
}
```

Especificar um atributo **`category`**

O processo de exportação infere automaticamente atributos `auto` para propriedades de string e propriedades numéricas que contenham vários valores. Para propriedades numéricas que contenham valores únicos, ele infere atributos `numerical`. Para propriedades de data, ele infere atributos `datetime`.

Se você quiser substituir uma especificação de atributo para que uma propriedade seja tratada como um atributo categórico, adicione `"type": "category"` à matriz de atributos. Se a propriedade contiver vários valores, inclua um campo `separator`. Por exemplo:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Post",
        "property": "tag",
        "type": "category",
        "separator": "|"
      }
    ]
  }
}
```

Especificar um atributo **`numerical`**

O processo de exportação infere automaticamente atributos `auto` para propriedades de string e propriedades numéricas que contenham vários valores. Para propriedades numéricas que contenham valores únicos, ele infere atributos `numerical`. Para propriedades de data, ele infere atributos `datetime`.

Se você quiser substituir uma especificação de atributo para que uma propriedade seja tratada como um atributo `numerical`, adicione `"type": "numerical"` à matriz de atributos. Se a propriedade contiver vários valores, inclua um campo `separator`. Por exemplo:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Recording",
        "property": "duration",
        "type": "numerical",
        "separator": ","
      }
    ]
  }
}
```

Especificar um atributo **`auto`**

O processo de exportação infere automaticamente atributos `auto` para propriedades de string e propriedades numéricas que contenham vários valores. Para propriedades numéricas que contenham valores únicos, ele infere atributos `numerical`. Para propriedades de data, ele infere atributos `datetime`.

Se você quiser substituir uma especificação de atributo para que uma propriedade seja tratada como um atributo `auto`, adicione `"type": "auto"` à matriz de atributos. Se a propriedade contiver vários valores, inclua um campo `separator`. Por exemplo:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "User",
        "property": "role",
```

```
        "type": "auto",
        "separator": ","
      }
    ]
  }
}
```

Exemplos do RDF usando **`additionalParams`**

Especificar uma taxa de divisão padrão para a configuração de treinamento de modelos

No exemplo a seguir, o parâmetro `split_rate` define a taxa de divisão padrão para o treinamento de modelos. Se nenhuma taxa de divisão padrão for especificada, o treinamento usará um valor de [0,9, 0,1, 0,0]. É possível substituir o valor padrão por destino especificando uma `split_rate` para cada destino.

No seguinte exemplo, o campo `default split_rate` indica que uma taxa de divisão de `[0.7,0.1,0.2]` deve ser usada, a menos que seja substituída por destino:”

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "split_rate": [0.7,0.1,0.2],
    "targets": [
      (...)
    ]
  }
}
```

Especificar uma tarefa de classificação de nós para configuração de treinamento de modelos

Para indicar qual propriedade de nó contém exemplos rotulados para fins de treinamento, adicione um elemento de classificação de nós à matriz `targets`, usando `"type" : "classification"`. Adicione um campo de nó para indicar o tipo dos nós de destino. Adicione um campo `predicate` para definir quais dados literais são usados como o atributo do nó de destino. Adicione um campo `split_rate` se quiser substituir a taxa de divisão padrão.

No exemplo a seguir, o destino `node` indica que a propriedade `genre` de cada nó `Movie` deve ser tratada como um rótulo de classe de nó. O valor `split_rate` substitui a taxa de divisão padrão:

```
  "additionalParams": {
```

```
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "node": "http://aws.amazon.com/neptune/csv2rdf/class/Movie",
        "predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/genre",
        "type": "classification",
        "split_rate": [0.7,0.1,0.2]
      }
    ]
  }
}
```

Especificar uma tarefa de regressão de nós para configuração de treinamento de modelos

Para indicar qual propriedade de nó contém regressões rotuladas para fins de treinamento, adicione um elemento de regressão de nós à matriz de destinos, usando `"type" : "regression"`. Adicione um campo `node` para indicar o tipo dos nós de destino. Adicione um campo `predicate` para definir quais dados literais são usados como o atributo do nó de destino. Adicione um campo `split_rate` se quiser substituir a taxa de divisão padrão.

O seguinte destino `node` indica que a propriedade `rating` de cada nó `Movie` deve ser tratada como um rótulo de regressão de nós:

```
    "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "node": "http://aws.amazon.com/neptune/csv2rdf/class/Movie",
        "predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/rating",
        "type": "regression",
        "split_rate": [0.7,0.1,0.2]
      }
    ]
  }
}
```

Especificar uma tarefa de previsão de links para bordas específicas

Para indicar quais bordas devem ser usadas para fins de treinamento de previsão de links, adicione um elemento de borda à matriz de destinos usando `"type" : "link_prediction"`. Adicione

os campos `subject`, `predicate` e `object` para especificar o tipo de borda. Adicione um campo `split_rate` se quiser substituir a taxa de divisão padrão.

O seguinte destino `edge` indica que as bordas `directed` que conectam `Directors` a `Movies` devem ser usadas para previsão de links:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "subject": "http://aws.amazon.com/neptune/csv2rdf/class/Director",
        "predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/directed",
        "object": "http://aws.amazon.com/neptune/csv2rdf/class/Movie",
        "type" : "link_prediction"
      }
    ]
  }
}
```

Especificar uma tarefa de previsão de links para todas as bordas

Para indicar que todas as bordas devem ser usadas para fins de treinamento de previsão de links, adicione um elemento `edge` à matriz de destinos usando `"type" : "link_prediction"`. Não adicione campos `subject`, `predicate` nem `object`. Adicione um campo `split_rate` se quiser substituir a taxa de divisão padrão.

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "type" : "link_prediction"
      }
    ]
  }
}
```

Processar os dados de grafos exportados do Neptune para treinamento

A etapa de processamento de dados usa os dados de grafos do Neptune criados pelo processo de exportação e cria as informações que são usadas pela Deep Graph Library (DGL) durante o treinamento. Isso inclui a realização de vários mapeamentos e transformações de dados:

- Análise de nós e bordas para criar os arquivos de mapeamento de grafos e de ID exigidos pela DGL.
- Conversão das propriedades do nó e da borda nos atributos do nó e da borda exigidos pela DGL.
- Divisão dos dados em conjuntos de treinamento, validação e teste.

Gerenciamento da etapa de processamento de dados do Neptune ML.

Depois de exportar os dados do Neptune que você deseja usar para treinamento de modelos, é possível iniciar um trabalho de processamento de dados usando um comando `curl` (ou `awscurl`) como o seguinte:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/dataprocessing \
  -H 'Content-Type: application/json' \
  -d '{
        "inputDataS3Location" : "s3://(Amazon S3 bucket name)/(path to your input
 folder)",
        "id" : "(a job ID for the new job)",
        "processedDataS3Location" : "s3://(S3 bucket name)/(path to your output
 folder)",
        "configFileName" : "training-job-configuration.json"
      }'
```

Os detalhes de como usar esse comando são explicados em O comando de processamento de dados, junto com informações sobre como obter o status de um trabalho em execução, como interrompê-lo e como listar todos os trabalhos em execução.

Processar dados de grafos atualizados para o Neptune ML

Também é possível fornecer um `previousDataProcessingJobId` à API para garantir que o novo trabalho de processamento de dados use o mesmo método de processamento de um trabalho

anterior. Esse procedimento é necessário quando você deseja obter previsões para dados de grafos atualizados no Neptune, seja treinando novamente o modelo antigo nos novos dados ou recalculando os artefatos do modelo nos novos dados.

Faça isso usando um comando `curl` (ou `awscurl`) desta forma:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/dataprocessing \
  -H 'Content-Type: application/json' \
  -d '{ "inputDataS3Location" : "s3://(Amazon S3 bucket name)/(path to your input
 folder)",
        "id" : "(a job ID for the new job)",
        "processedDataS3Location" : "s3://(Amazon S3 bucket name)/(path to your output
 folder)",
        "previousDataProcessingJobId", "(the job ID of the previous data-processing
 job)"}'
```

Defina o valor do parâmetro `previousDataProcessingJobId` como o ID de trabalho do processamento de dados anterior correspondente ao modelo treinado.

> **Note**
>
> No momento, exclusões de nós no grafo atualizado não são aceitas. Se os nós tiverem sido removidos em um grafo atualizado, você precisará iniciar um trabalho de processamento de dados completamente novo em vez de usar `previousDataProcessingJobId`.

Codificação de atributos no Neptune ML

Os valores das propriedades têm diferentes formatos e tipos de dados. Para obter um bom desempenho no machine learning, é essencial converter esses valores em codificações numéricas conhecidas como atributos.

O Neptune ML realiza extração e codificação de atributos como parte das etapas de exportação e processamento de dados, usando técnicas de codificação de atributos descritas aqui.

> **Note**
>
> Se você planeja implementar a própria codificação de atributos em uma implementação de modelo personalizado, é possível desabilitar a codificação automática de atributos na fase de pré-processamento de dados selecionando `none` como o tipo de codificação de atributos. Nenhuma codificação de atributos ocorre nessa propriedade de nó ou borda e, em vez disso, os valores brutos da propriedade são analisados e salvos em um dicionário. O pré-processamento de dados ainda cria o grafo DGL a partir do conjunto de dados exportado, mas o grafo DGL criado não tem os atributos pré-processados para treinamento.
> Você deve usar essa opção se planeja realizar a própria codificação de atributos personalizados como parte do treinamento de modelos personalizados. Para obter detalhes, consulte Modelos personalizados no Neptune ML.

Atributos categóricos no Neptune ML

Uma propriedade que pode receber um ou mais valores distintos de uma lista fixa de valores possíveis é um atributo categórico. No Neptune ML, os atributos categóricos são codificados usando a codificação one-hot. O seguinte exemplo mostra como o nome da propriedade de diferentes alimentos é codificado como one-hot de acordo com a categoria:

```
 Food        Veg.   Meat   Fruit    Encoding
---------    ----   ----   -----    --------
 Apple         0      0      1         001
 Chicken       0      1      0         010
 Broccoli      1      0      0         100
```

> **Note**
>
> O número máximo de categorias em qualquer atributo categórico é cem. Se uma propriedade tiver mais de cem categorias de valor, somente as 99 mais comuns serão colocadas em categorias distintas e as demais serão colocadas em uma categoria especial chamada `OTHER`.

Atributos numéricos no Neptune ML

Qualquer propriedade cujos valores sejam números reais pode ser codificada como um atributo numérico no Neptune ML. Os atributos numéricos são codificados usando números de ponto flutuante.

É possível especificar um método de normalização de dados a ser usado ao codificar atributos numéricos, como este: `"norm": "normalization technique"` As seguintes técnicas de normalização são compatíveis:

- “nenhum”: não normalize os valores numéricos durante a codificação.
- “mín-máx”: normalize cada valor subtraindo o valor mínimo e dividindo-o pela diferença entre o valor máximo e o mínimo.
- “padrão”: normalize cada valor dividindo-o pela soma de todos os valores.

Atributos numéricos de bucket no Neptune ML

Em vez de representar uma propriedade numérica usando números brutos, é possível condensar valores numéricos em categorias. Por exemplo, é possível dividir a idade das pessoas em categorias como crianças (0 a 20 anos), jovens adultos (20 a 40), pessoas de meia idade (40 a 60 anos) e idosos (de 60 anos em diante). Usando esses buckets numéricos, você transformaria uma propriedade numérica em um tipo de atributo categórico.

No Neptune ML, é possível fazer com que uma propriedade numérica seja codificada como um atributo numérico de bucket. Você deve fornecer dois itens:

- Um intervalo numérico no formato, `"range": [a, b]` , em que a e b são números inteiros.
- Uma contagem de buckets, no formato `"bucket_cnt": c` , em que `c` é o número de buckets, também um número inteiro.

Depois, o Neptune ML calcula o tamanho de cada bucket como `( b - a ) / c` e codifica cada valor numérico como o número de qualquer bucket em que ele se enquadra. Qualquer valor menor que a é considerado pertencente ao primeiro bucket, e qualquer valor maior que b é considerado pertencente ao último bucket.

Você também pode fazer com que os valores numéricos se enquadrem em mais de um bucket, especificando um tamanho de janela deslizante, como este : `"slide_window_size": s` , em que s é um número. Depois, o Neptune ML transforma cada valor numérico v da propriedade em um intervalo de `v - s/2` a `v + s/2` e atribui o valor v a cada bucket coberto pelo intervalo.

Por fim, também é possível fornecer uma forma de preencher valores ausentes para atributos numéricos e atributos numéricos de bucket. Você faz isso usando `"imputer": "imputation technique "`, em que a técnica de imputação é uma das seguintes: `"mean"`, `"median"` ou `"most-frequent"`. Se não for especificado um estimador, um valor ausente poderá fazer com que o processamento seja interrompido.

Codificação de atributos de texto no Neptune ML

Para texto de formato livre, o Neptune ML pode usar vários modelos diferentes para converter a sequência de tokens em uma string de valor de propriedade em um vetor de valor real de tamanho fixo:

- `text_fasttext`: usa a codificação fastText. É a codificação recomendada para atributos que usam um e somente um dos cinco idiomas aceitos pela codificação fastText.
- `text_sbert`: usa os modelos de codificação Sentence BERT (SBERT). É a codificação recomendada para texto que não é compatível com `text_fasttext`.
- `text_word2vec`: usa os algoritmos Word2Vec originalmente publicados pelo Google para codificar texto. O Word2Vec é compatível apenas com inglês.
- `text_tfidf`: usa um vetorizador de frequência de termo – frequência inversa do documento (TF-IDF) para codificar texto. A codificação TF-IDF é compatível com atributos estatísticos não aceitos por outras codificações.

Codificação fastText de valores de propriedades de texto no Neptune ML

O Neptune ML pode usar os modelos de fastText para converter valores de propriedades de texto em vetores de valor real de tamanho fixo. Esse é o método de codificação recomendado para valores de propriedades de texto em qualquer um dos cinco idiomas compatíveis com o fastText:

- `en` (inglês)
- `zh` (chinês)
- `hi` (hindi)
- `es` (espanhol)
- `fr` (francês)

Observe que a codificação fastText não consegue lidar com palavras em mais de um idioma.

O método `text_fasttext` pode usar o campo `max_length` que especifica o número máximo de tokens em um valor de propriedade de texto que será codificado, após o qual a string será truncada. Isso pode melhorar o desempenho quando os valores das propriedades de texto contêm strings longas, porque, se `max_length` não for especificado, a fastText codificará todos os tokens, independentemente do tamanho da string.

Este exemplo especifica que os títulos de filmes franceses são codificados usando fastText:

```
{
    "file_name" : "nodes/movie.csv",
    "separator" : ",",
    "node" : ["~id", "movie"],
    "features" : [
      {
        "feature": ["title", "title", "text_fasttext"],
        "language": "fr",
        "max_length": 1024
      }
    ]
  }
```

Codificação Sentence BERT (SBERT) de atributos de texto no Neptune ML

O Neptune ML pode converter a sequência de tokens de um valor de propriedade de string em um vetor de valor real de tamanho fixo usando modelos de Sentence BERT (SBERT). O Neptune é compatível com dois métodos de SBERT, ou seja, `text_sbert128`, que será o padrão se você especificar apenas `text_sbert` e `text_sbert512`. A diferença entre os dois é o tamanho máximo de uma string de valor de propriedade de texto codificada. A codificação `text_sbert128` trunca strings de texto depois de codificar 128 tokens, enquanto `text_sbert512` trunca strings de texto depois de codificar 512 tokens. Como resultado, `text_sbert512` pode exigir mais

tempo de processamento do que `text_sbert128`. Os dois métodos são mais lentos do que `text_fasttext`.

A codificação SBERT é multilíngue, portanto, não há necessidade de especificar um idioma para o texto do valor da propriedade que você está codificando. O SBERT é compatível com vários idiomas e pode codificar uma frase que contenha mais de um idioma. Se você estiver codificando valores de propriedades que contenham texto em um idioma ou idiomas que o fastText não aceita, o SBERT será o método de codificação recomendado.

O exemplo a seguir especifica que os títulos de filmes sejam codificados como SBERT até no máximo 128 tokens:

```
{
    "file_name" : "nodes/movie.csv",
    "separator" : ",",
    "node" : ["~id", "movie"],
    "features" : [
      { "feature": ["title", "title", "text_sbert128"] }
    ]
  }
```

Codificação Word2Vec de atributos de texto no Neptune ML

O Neptune ML pode codificar valores de propriedades de string como um atributo do Word2Vec (atributos Word2Vec foram publicados originalmente pelo Google). O método `text_word2vec` codifica os tokens em uma string como um vetor denso usando um dos modelos treinados pelo spaCy. Isso é compatível apenas com o idioma inglês usando o modelo en_core_web_lg).

O seguinte exemplo especifica que os títulos dos filmes sejam codificados usando Word2Vec:

```
{
    "file_name" : "nodes/movie.csv",
    "separator" : ",",
    "node" : ["~id", "movie"],
    "features" : [
      {
        "feature": ["title", "title", "text_word2vec"],
        "language": "en_core_web_lg"
      }
    ]
  }
```

Observe que o campo de idioma é opcional, pois o modelo `en_core_web_lg` em inglês é o único compatível com o Neptune.

Codificação TF-IDF de atributos de texto no Neptune ML

O Neptune ML pode codificar valores de propriedades de texto como atributos `text_tfidf`. Essa codificação converte a sequência de palavras no texto em um vetor numérico usando um vetorizador de frequência de termo – frequência inversa de documento (TF-IDF), seguido por uma operação de redução de dimensionalidade.

TF-IDF (frequência de termo – frequência inversa de documento) é um valor numérico destinado a medir a importância de uma palavra em um conjunto de documentos. É calculado dividindo o número de vezes que uma palavra aparece em um valor de propriedade específico pelo número total desses valores de propriedade em que ela aparece.

Por exemplo, se a palavra “kiss” aparecer duas vezes em um título de filme específico (digamos, “kiss kiss bang bang”) e “kiss” aparecer no título de quatro filmes ao todo, o valor TF-IDF de “kiss” no título “kiss kiss bang bang” será `2 / 4` .

O vetor criado inicialmente tem dimensões d, em que d é o número de termos exclusivos em todos os valores de propriedade desse tipo. A operação de redução de dimensionalidade usa uma projeção esparsa aleatória para reduzir esse número a um máximo de cem. O vocabulário de um grafo é então gerado pela fusão de todos os atributos `text_tfidf` contidos nele.

É possível controlar o vetorizador TF-IDF de várias maneiras:

- **`max_features`**: usando o parâmetro `max_features`, é possível limitar o número de termos nos atributos `text_tfidf` aos mais comuns. Por exemplo, se você definir `max_features` como cem, somente os cem termos mais usados serão incluídos. O valor padrão para `max_features` se você não o definir explicitamente será 5 mil.
- **`min_df`**: usando o parâmetro `min_df`, você poderá limitar o número de termos nos atributos `text_tfidf` àqueles que tenham pelo menos uma frequência de documento especificada. Por exemplo, se você definir `min_df` como cinco, somente os termos exibidos em pelo menos cinco valores de propriedade diferentes serão usados. O valor padrão para `min_df` se você não o definir explicitamente será dois.
- **`ngram_range`**: o parâmetro `ngram_range` determina quais combinações de palavras são tratadas como termos. Por exemplo, se você definir `ngram_range` como `[2, 4]`, os seguintes seis termos serão encontrados no título “kiss kiss bang bang”:

- Termos de duas palavras: “kiss kiss”, “kiss bang” e “bang bang”.
- Termos de três palavras: “kiss kiss bang” e “kiss bang bang”.
- Termos de quatro palavras: “kiss kiss bang bang”.

A configuração padrão para `ngram_range` é `[1, 1]`.

Atributos de data e hora no Neptune ML

O Neptune ML pode converter partes de valores de propriedades `datetime` em atributos categóricos codificando-os como matrizes one-hot. Use o parâmetro `datetime_parts` para especificar uma ou mais das seguintes partes a serem codificadas: `["year", "month", "weekday", "hour"]`. Se você não definir `datetime_parts`, por padrão, todas as quatro partes serão codificadas.

Por exemplo, se o intervalo de valores de data e hora abranger os anos de 2010 a 2012, as quatro partes da entrada de data e hora `2011-04-22 01:16:34` serão as seguintes:

- ano: `[0, 1, 0]`.

  Como há apenas três anos no período (2010, 2011 e 2012), a matriz one-hot tem três entradas, uma para cada ano.
- mês: `[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]`.

  Aqui, a matriz one-hot tem uma entrada para cada mês do ano.
- dia da semana: `[0, 0, 0, 0, 1, 0, 0]`.

  O padrão ISO 8601 estabelece que segunda-feira é o primeiro dia da semana e, como 22 de abril de 2011 foi sexta-feira, a matriz de dia da semana one-hot correspondente está ativa na quinta posição.
- hora: `[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]`.

  A hora da 1h é definida em uma matriz one-hot de 24 membros.

Dia do mês, minuto e segundo não são codificados categoricamente.

Se o intervalo `datetime` total em questão incluir apenas datas de um único ano, nenhuma matriz `year` será codificada.

É possível especificar uma estratégia de imputação para preencher os valores `datetime` ausentes, usando o parâmetro `imputer` e uma das estratégias disponíveis para atributos numéricos.

Codificação de atributos automáticos no Neptune ML

Em vez de especificar manualmente os métodos de codificação de atributos a serem usados para as propriedades em seu grafo, você pode definir `auto` como um método de codificação de atributos. Depois, o Neptune ML tenta inferir a melhor codificação de atributos para cada propriedade com base no tipo de dados subjacente.

Veja algumas das heurísticas que o Neptune ML usa para selecionar as codificações de atributos apropriadas:

- Se a propriedade tiver somente valores numéricos e puder ser convertida em tipos de dados numéricos, o Neptune ML geralmente a codificará como um valor numérico. No entanto, se o número de valores exclusivos da propriedade for menor que 10% do número total de valores e a cardinalidade desses valores exclusivos for menor que cem, o Neptune ML usará uma codificação categórica.
- Se os valores das propriedades puderem ser convertidos em um tipo `datetime`, o Neptune ML os codificará como um atributo `datetime`.
- Se os valores das propriedades puderem ser forçados a boolianos (1/0 ou verdadeiro/falso), o Neptune ML usará a codificação de categorias.
- Se a propriedade for uma string com mais de 10% de seus valores exclusivos e o número médio de tokens por valor for maior ou igual a três, o Neptune ML vai inferir que o tipo de propriedade é texto e detectará automaticamente o idioma que está sendo usado. Se o idioma detectado for um dos aceitos pelo fastText, ou seja, inglês, chinês, hindi, espanhol e francês, o Neptune ML usará `text_fasttext` para codificar o texto . Caso contrário, o Neptune ML usará `text_sbert`.
- Se a propriedade for uma string não classificada como um atributo de texto, o Neptune ML presumirá que seja um atributo categórico e usará a codificação de categoria.
- Se cada nó tiver o próprio valor exclusivo para uma propriedade inferida como um atributo de categoria, o Neptune ML vai retirar a propriedade do grafo de treinamento porque provavelmente é um ID que não seria informativo para aprendizado.
- Se a propriedade contiver separadores válidos do Neptune, como ponto e vírgula (";"), o Neptune ML só poderá tratar a propriedade como `MultiNumerical` ou `MultiCategorical`.
  - O Neptune ML primeiro tenta codificar os valores como atributos numéricos. Se isso for bem-sucedido, o Neptune ML usará codificação numérica para criar atributos de vetor numérico.

  - Caso contrário, o Neptune ML codificará os valores como multicategóricos.
- Se o Neptune ML não puder inferir o tipo de dados dos valores de uma propriedade, o Neptune ML eliminará a propriedade do grafo de treinamento.

Editar um arquivo de configuração de dados de treinamento

O processo de exportação do Neptune exporta dados do Neptune ML de um cluster de banco de dados do Neptune para um bucket do S3. Ele exporta nós e bordas separadamente para uma pasta `nodes/` e uma `edges/`. Ele também cria um arquivo de configuração de dados de treinamento JSON, denominado `training-data-configuration.json` por padrão. Esse arquivo contém informações sobre o esquema do grafo, os tipos de atributos, as operações de transformação e normalização de atributos e o atributo de destino para uma tarefa de classificação ou regressão.

Pode haver casos em que você queira modificar o arquivo de configuração diretamente. Um desses casos é quando você deseja alterar a forma como os atributos são processados ou como o grafo é criado, sem precisar executar novamente a exportação toda vez que quiser modificar a especificação da tarefa de machine learning que você está resolvendo.

Como editar o arquivo de configuração de dados de treinamento

1. Baixe o arquivo na máquina local.

   A menos que você tenha especificado um ou mais trabalhos nomeados no parâmetro `additionalParams/neptune_ml` transmitido para o processo de exportação, o arquivo terá o nome padrão, que é `training-data-configuration.json`. É possível usar um comando AWS da CLI como este para baixar o arquivo:

   ```
   aws s3 cp \
     s3://(your Amazon S3 bucket)/(path to your export folder)/training-data-
   configuration.json \
     ./
   ```

2. Edite o arquivo usando um editor de texto.
3. Faça upload do arquivo modificado. Faça upload do arquivo modificado de volta para o mesmo local no Amazon S3 do qual você o baixou, usando um comando AWS da CLI como este:

   ```
   aws s3 cp \
     training-data-configuration.json \
     s3://(your Amazon S3 bucket)/(path to your export folder)/training-data-
   configuration.json
   ```

Exemplo de arquivo de configuração de dados de treinamento JSON

Veja um exemplo de arquivo de configuração de dados de treinamento que descreve um grafo para uma tarefa de classificação de nós:

```
{
  "version" : "v2.0",
  "query_engine" : "gremlin",
  "graph" : [
    {
      "edges" : [
        {
          "file_name" : "edges/(movie)-included_in-(genre).csv",
          "separator" : ",",
          "source" : ["~from", "movie"],
          "relation" : ["", "included_in"],
          "dest" : [ "~to", "genre" ]
        },
        {
          "file_name" : "edges/(user)-rated-(movie).csv",
          "separator" : ",",
          "source" : ["~from", "movie"],
          "relation" : ["rating", "prefixname"], # [prefixname#value]
          "dest" : ["~to", "genre"],
          "features" : [
            {
              "feature" : ["rating", "rating", "numerical"],
              "norm" : "min-max"
            }
          ]
        }
      ],
      "nodes" : [
        {
          "file_name" : "nodes/genre.csv",
          "separator" : ",",
          "node" : ["~id", "genre"],
          "features" : [
            {
              "feature": ["name", "genre", "category"],
              "separator": ";"
            }
          ]
```

```
        },
        {
          "file_name" : "nodes/movie.csv",
          "separator" : ",",
          "node" : ["~id", "movie"],
          "features" : [
            {
              "feature": ["title", "title", "word2vec"],
              "language": ["en_core_web_lg"]
            }
          ]
        },
        {
          "file_name" : "nodes/user.csv",
          "separator" : ",",
          "node" : ["~id", "user"],
          "features" : [
            {
              "feature": ["age", "age", "numerical"],
              "norm" : "min-max",
              "imputation": "median",
            },
            {
              "feature": ["occupation", "occupation", "category"],
            }
          ],
          "labels" : [
            {
              "label": ["gender", "classification"],
              "split_rate" : [0.8, 0.2, 0.0]
            }
          ]
        }
      ]
    },
    "warnings" : [ ]
  ]
}
```

A estrutura dos arquivos de configuração de dados de treinamento JSON

O arquivo de configuração de treinamento se refere aos arquivos CSV salvos pelo processo de exportação nas pastas `nodes/` e `edges/`.

Cada arquivo em `nodes/` armazena informações sobre nós que têm o mesmo rótulo de nó do grafo de propriedades. Cada coluna em um arquivo de nó armazena o ID ou a propriedade do nó. A primeira linha do arquivo contém um cabeçalho que especifica o `~id` ou o nome da propriedade de cada coluna.

Cada arquivo em `edges/` armazena informações sobre nós que têm o mesmo rótulo de borda do grafo de propriedades. Cada coluna em um arquivo de nó armazena o ID do nó de origem, o ID do nó de destino ou a propriedade da borda. A primeira linha do arquivo contém um cabeçalho que especifica `~from`, `~to` ou o nome da propriedade de cada coluna.

O arquivo de configuração de dados de treinamento tem três elementos gerais:

```
{
  "version" : "v2.0",
  "query_engine" : "gremlin",
  "graph" : [ ... ]
}
```

- `version`: (string) a versão do arquivo de configuração que está sendo usado.
- `query_engine`: (string) a linguagem de consulta usada para exportar os dados do grafo. No momento, somente “gremlin” é válido.
- `graph`: (matriz JSON) lista um ou mais objetos de configuração que contêm parâmetros de modelo para cada um dos nós e bordas que serão usados.

  Os objetos de configuração na matriz de grafos têm a estrutura descrita na próxima seção.

Conteúdo de um objeto de configuração listado na matriz **`graph`**

Um objeto de configuração na matriz `graph` pode conter três nós gerais:

```
    {
      "edges"    : [ ... ],
      "nodes"    : [ ... ],
      "warnings" : [ ... ],
    }
```

- `edges`: (matriz de objetos JSON) cada objeto JSON especifica um conjunto de parâmetros para definir como uma borda no grafo será tratada durante o processamento e o treinamento de modelos. Isso só é usado com o mecanismo do Gremlin.

- `nodes`: (matriz de objetos JSON) cada objeto JSON especifica um conjunto de parâmetros para definir como um nó no grafo será tratado durante o processamento e o treinamento de modelos. Isso só é usado com o mecanismo do Gremlin.
- `warnings`: (matriz de objetos JSON) cada objeto contém um aviso gerado durante o processo de exportação de dados.

Conteúdo de um objeto de configuração de borda listado na matriz **`edges`**

Um objeto de configuração de borda listado em uma matriz `edges` pode conter os seguintes campos gerais:

```
    {
      "file_name" : "(path to a CSV file)",
      "separator" : "(separator character)",
      "source"    : ["(column label for starting node ID)", "(starting node type)"],
      "relation"  : ["(column label for the relationship name)", "(the prefix name
for the relationship name)"],
      "dest"      : ["(column label for ending node ID)", "(ending node type)"],
      "features"  : [(array of feature objects)],
      "labels"    : [(array of label objects)]
    }
```

- **`file_name`**: uma string especificando o caminho para um arquivo CSV que armazena informações sobre bordas com o mesmo rótulo de grafo de propriedades.

  A primeira linha desse arquivo contém uma linha de cabeçalho de rótulos de coluna.

  Os dois primeiros rótulos de coluna são `~from` e `~to`. A primeira coluna (`~from`) armazena o ID do nó inicial da borda e a segunda (`~to`) armazena o ID do nó final da borda.

  Os rótulos das colunas restantes na linha do cabeçalho especificam, para cada coluna restante, o nome da propriedade da borda cujos valores foram exportados para essa coluna.
- **`separator`**: uma string contendo o delimitador que separa as colunas nesse arquivo CSV.
- **`source`**: uma matriz JSON contendo duas strings que especificam o nó inicial da borda. A primeira string contém o nome do cabeçalho da coluna na qual o ID do nó inicial está armazenado. A segunda string especifica o tipo de nó.

- **`relation`**: uma matriz JSON contendo duas strings que especificam o tipo de relação da borda. A primeira string contém o nome do cabeçalho da coluna na qual o nome da relação (`relname`) está armazenado. A segunda string contém o prefixo para o nome da relação (`prefixname`).

  O tipo de relação completo consiste em duas strings combinadas, com um caractere de hífen entre elas, como: *`prefixname`*`-`*`relname`*.

  Se a primeira string estiver vazia, todas as bordas terão o mesmo tipo de relação, ou seja, a string `prefixname`.
- **`dest`**: uma matriz JSON contendo duas strings que especificam o nó final da borda. A primeira string contém o nome do cabeçalho da coluna na qual o ID do nó está armazenado. A segunda string especifica o tipo de nó.
- **`features`**: uma matriz JSON de objetos de atributos de valor de propriedade. Cada objeto de atributo de valor de propriedade contém os seguintes campos:
  - atributo: uma matriz JSON de três strings. A primeira string contém o nome do cabeçalho da coluna que contem o valor de propriedade. A segunda string contém o nome do atributo. A terceira string contém o tipo de atributo.
  - norma: (opcional) especifica um método de normalização a ser aplicado aos valores da propriedade.

- **`labels`**: uma matriz JSON de objetos. Cada um dos objetos define um atributo de destino das bordas e especifica as proporções das bordas que as fases de treinamento e validação devem assumir. Cada objeto contém os seguintes campos:
  - rótulo: uma matriz JSON de duas strings. A primeira string contém o nome do cabeçalho da coluna que contem o valor de propriedade do atributo de destino. A segunda string especifica um dos seguintes tipos de tarefa de destino:
    - `"classification"`: uma tarefa de classificação de borda. Os valores das propriedades fornecidos na coluna identificada pela primeira string na matriz `label` são tratados como valores categóricos. Para uma tarefa de classificação de bordas, a primeira string na matriz `label` não pode estar vazia.
    - `"regression"`: uma tarefa de regressão de bordas. Os valores das propriedades fornecidos na coluna identificada pela primeira string na matriz `label` são tratados como valores numéricos. Para uma tarefa de regressão de bordas, a primeira string na matriz `label` não pode estar vazia.

  - `"link_prediction"`: uma tarefa de previsão de links. Nenhum valor de propriedade é obrigatório. Para uma tarefa de previsão de links, a primeira string na matriz `label` é ignorada.
- **`split_rate`**: uma matriz JSON contendo três números entre zero e um que somam um e representam uma estimativa das proporções de nós que as fases de treinamento, validação e teste usarão, respectivamente. Esse campo ou `custom_split_filenames` podem ser definidos, mas não ambos. Veja split_rate.
- **`custom_split_filenames`**: um objeto JSON que especifica os nomes dos arquivos que definem as populações de treinamento, validação e teste. Esse campo ou `split_rate` podem ser definidos, mas não ambos. Consulte Proporções personalizadas de treinamento, validação e teste para obter mais informações.

Conteúdo de um objeto de configuração de nó listado na matriz **`nodes`**

Um objeto de configuração de nó listado em uma matriz `nodes` pode conter os seguintes campos:

```
{
  "file_name" : "(path to a CSV file)",
  "separator" : "(separator character)",
  "node"      : ["(column label for the node ID)", "(node type)"],
  "features"  : [(feature array)],
  "labels"    : [(label array)],
}
```

- **`file_name`**: uma string especificando o caminho para um arquivo CSV que armazena informações sobre nó com o mesmo rótulo de grafo de propriedades.

  A primeira linha desse arquivo contém uma linha de cabeçalho de rótulos de coluna.

  O rótulo da primeira coluna é `~id`, e a primeira coluna (`~id`) armazena o ID do nó.

  Os rótulos das colunas restantes na linha do cabeçalho especificam, para cada coluna restante, o nome da propriedade do nó cujos valores foram exportados para essa coluna.
- **`separator`**: uma string contendo o delimitador que separa as colunas nesse arquivo CSV.
- **`node`**: uma matriz JSON contendo duas strings. A primeira string contém o nome do cabeçalho da coluna que armazena os IDs dos nós. A segunda string especifica o tipo de nó no grafo, que corresponde a um rótulo de grafo de propriedades do nó.

- **`features`**: uma matriz JSON de objetos de atributos de nós. Consulte Conteúdo de um objeto de atributo listado em uma matriz `features` para um nó ou uma borda.
- **`labels`**: uma matriz JSON de objetos de rótulos de nós. Consulte Conteúdo de um objeto de rótulo de nó listado em uma matriz `labels` de nós.

Conteúdo de um objeto de atributo listado em uma matriz **`features`** para um nó ou uma borda

Um objeto de atributo de nó listado em uma matriz `features` de nó pode conter os seguintes campos gerais:

- **`feature`**: uma matriz JSON de três strings. A primeira string contém o nome do cabeçalho da coluna que contem o valor de propriedade do atributo. A segunda string contém o nome do atributo.

  A terceira string contém o tipo de atributo. Os tipos de atributos válidos estão listados em Valores possíveis do campo `type` para atributos.
- **`norm`**: esse campo é obrigatório para atributos numéricos. Ele especifica um método de normalização a ser usado em valores numéricos. Os valores válidos são `"none"`, `"min-max"` e "padrão". Para obter detalhes, consulte O campo `norm`.
- **`language`**: o campo de idioma especifica o idioma usado nos valores das propriedades de texto. O uso depende do método de codificação de texto:
  - Para a codificação `text_fasttext`, esse campo é obrigatório e deve especificar um dos seguintes idiomas:
    - `en` (inglês)
    - `zh` (chinês)
    - `hi` (hindi)
    - `es` (espanhol)
    - `fr` (francês)

    No entanto, `text_fasttext` não consegue lidar com mais de um idioma por vez.
  - Para a codificação `text_sbert`, esse campo não é usado, pois a codificação SBERT é multilíngue.
  - Para a codificação `text_word2vec`, esse campo é opcional, pois `text_word2vec` só é compatível com inglês. Se presente, ele deve especificar o nome do modelo em inglês:

```
"language" : "en_core_web_lg"
```

  - Para a codificação `tfidf`, esse campo não é usado.

- **`max_length`**: esse campo é opcional para atributos `text_fasttext`, no qual ele especifica o número máximo de tokens em um atributo de texto de entrada que será codificado. O texto de entrada após `max_length` ser atingido é ignorado. Por exemplo, definir max_length como 128 indica que qualquer token após o 128º em uma sequência de texto será ignorado.
- **`separator`**: esse campo é usado opcionalmente com os atributos `category`, `numerical` e `auto`. Ele especifica um caractere que pode ser usado para dividir o valor de uma propriedade em vários valores categóricos ou valores numéricos.

  Consulte O campo `separator`.
- **`range`**: esse campo é obrigatório para atributos `bucket_numerical`. Ele especifica o intervalo de valores numéricos que devem ser divididos em buckets.

  Consulte O campo `range`.
- **`bucket_cnt`**: esse campo é obrigatório para atributos `bucket_numerical`. Ele especifica o número de buckets nos quais o intervalo numérico definido pelo parâmetro `range` deve ser dividido:

  Consulte Atributos numéricos de bucket no Neptune ML.
- **`slide_window_size`**: esse campo é usado opcionalmente com atributos `bucket_numerical` para atribuir valores a mais de um bucket:

  Consulte O campo `slide_window_size`.
- **`imputer`**: esse campo é usado opcionalmente com atributos `numerical`, `bucket_numerical` e `datetime` para fornecer uma técnica de imputação a fim de preencher valores ausentes. As técnicas de imputação compatíveis são `"mean"`, `"median"` e `"most_frequent"`.

  Consulte O campo `imputer`.
- **`max_features`**: esse campo é usado opcionalmente pelos atributos `text_tfidf` para especificar o número máximo de termos a serem codificados.

  Consulte O campo `max_features`.
- **`min_df`**: esse campo é usado opcionalmente pelos atributos `text_tfidf` para especificar a frequência mínima de documentos de termos a serem codificados.

Consulte O campo `min_df`.

- **`ngram_range`**: esse campo é usado opcionalmente pelos atributos `text_tfidf` para especificar um intervalo de números de palavras ou tokens a serem considerados como possíveis termos individuais a serem codificados.

  Consulte O campo `ngram_range`.

- **`datetime_parts`**: esse campo é usado opcionalmente pelos atributos `datetime` para especificar quais partes do valor de data e hora devem ser codificadas categoricamente:

  Consulte O campo `datetime_parts`.

Conteúdo de um objeto de rótulo de nó listado em uma matriz **`labels`** de nós

Um objeto de rótulo listado em uma matriz `labels` de nós define um atributo de destino de nó e especifica as proporções dos nós que as fases de treinamento, validação e teste usarão. Cada objeto pode conter os seguintes campos:

```
    {
      "label"      : ["(column label for the target feature property value)", "(task
type)"],
      "split_rate" : [(training proportion), (validation proportion), (test
proportion)],
      "custom_split_filenames" : {"train": "(training file name)", "valid":
"(validation file name)", "test": "(test file name)"},
      "separator"  : "(separator character for node-classification category values)",
    }
```

- **`label`**: uma matriz JSON contendo duas strings. A primeira string contém o nome do cabeçalho da coluna que armazena os valores de propriedade do atributo. A segunda string especifica o tipo de tarefa de destino, que pode ser:
  - `"classification"`: uma tarefa de classificação de nós. Os valores das propriedades na coluna especificada são usados para criar um atributo categórico.
  - `"regression"`: uma tarefa de regressão de nós. Os valores das propriedades na coluna especificada são usados para criar um atributo numérico.
- **`split_rate`**: uma matriz JSON contendo três números entre zero e um que somam um e representam uma estimativa das proporções de nós que as fases de treinamento, validação e teste usarão, respectivamente. Consulte split_rate.

- **`custom_split_filenames`**: um objeto JSON que especifica os nomes dos arquivos que definem as populações de treinamento, validação e teste. Esse campo ou `split_rate` podem ser definidos, mas não ambos. Consulte Proporções personalizadas de treinamento, validação e teste para obter mais informações.
- **`separator`**: uma string contendo o delimitador que separa os valores de atributos categóricos de uma tarefa de classificação.

> **Note**
>
> Se nenhum objeto de rótulo for fornecido para bordas e nós, a tarefa será automaticamente considerada como previsão de links, e as bordas serão divididas aleatoriamente em 90% para treinamento e 10% para validação.

Proporções personalizadas de treinamento, validação e teste

Por padrão, o parâmetro `split_rate` é usado pelo Neptune ML para dividir o grafo aleatoriamente em populações de treinamento, validação e teste usando as proporções definidas nesse parâmetro. Para ter um controle mais preciso sobre quais entidades são usadas nessas diferentes populações, podem ser criados arquivos que as definam explicitamente e, depois, o arquivo de configuração de dados de treinamento pode ser editado para associar esses arquivos de indexação às populações. Esse mapeamento é especificado por um objeto JSON para a chave `custom_split_filesnames` no arquivo de configuração de treinamento. Se essa opção for usada, os nomes dos arquivos deverão ser fornecidos para as chaves `train` e `validation` e é opcional para a chave `test`.

A formatação desses arquivos deve corresponder ao formato de dados do Gremlin. Especificamente, para tarefas em nível de nó, cada arquivo deve conter uma coluna com o cabeçalho `~id` que lista os IDs dos nós e, para tarefas em nível de borda, os arquivos devem especificar `~from` e `~to` para indicar os nós de origem e destino das bordas, respectivamente. Esses arquivos precisam ser colocados no mesmo local do Amazon S3 que os dados exportados que são usados para processamento de dados (consulte: `outputS3Path`).

Para tarefas de classificação ou regressão de propriedades, esses arquivos podem definir os rótulos para a tarefa de machine learning. Nesse caso, os arquivos precisam ter uma coluna de propriedades com o mesmo nome de cabeçalho definido no arquivo de configuração de dados de treinamento. Se os rótulos de propriedades forem definidos nos arquivos de nó e borda exportados

e nos arquivos de divisão personalizada, a prioridade será fornecida aos arquivos de divisão personalizada.

Treinar um modelo usando o Neptune ML

Depois de exportar os dados exportados do Neptune para treinamento de modelos, é possível iniciar um trabalho de processamento de modelos usando um comando `curl` (ou `awscurl`) como o seguinte:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltraining
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer"
      }'
```

Os detalhes de como usar esse comando são explicados em O comando modeltraining, junto com informações sobre como obter o status de um trabalho em execução, como interrompê-lo e como listar todos os trabalhos em execução.

Você também pode fornecer um `previousModelTrainingJobId` para usar informações de um trabalho de treinamento de modelos do Neptune ML concluído para acelerar a pesquisa de hiperparâmetros em um novo trabalho de treinamento. Isso é útil durante o novo treinamento de modelos em novos dados de grafos, bem como no treinamento incremental nos mesmos dados de grafos. Use um comando como este:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltraining
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer"
        "previousModelTrainingJobId" : "(the model-training job-id of a completed job)"
      }'
```

É possível treinar sua própria implementação de modelos na infraestrutura de treinamento do Neptune ML fornecendo um objeto `customModelTrainingParameters`, como este:

```
curl \
```

```
  -X POST https://(your Neptune endpoint)/ml/modeltraining
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer"
        "modelName": "custom",
        "customModelTrainingParameters" : {
          "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your Python
 module)",
          "trainingEntryPointScript": "(your training script entry-point name in the
 Python module)",
          "transformEntryPointScript": "(your transform script entry-point name in the
 Python module)"
        }
      }'
```

Consulte O comando modeltraining para obter mais informações, por exemplo, sobre como obter o status de um trabalho em execução, como interrompê-lo e como listar todos os trabalhos em execução. Consulte Modelos personalizados no Neptune ML para obter informações sobre como implementar e usar um modelo personalizado.

Tópicos

- Modelos e treinamento de modelos no Amazon Neptune ML
- Personalizar as configurações de hiperparâmetros do modelo no Neptune ML
- Práticas recomendadas para treinamento de modelos

Modelos e treinamento de modelos no Amazon Neptune ML

O Neptune ML usa redes neurais para grafos (GNN) a fim de criar modelos para as várias tarefas de machine learning. Foi demonstrado que as redes neurais para grafos obtêm resultados de última geração para tarefas de machine learning para grafos e são excelentes para extrair padrões informativos de dados estruturados em grafos.

Redes neurais para grafos (GNNs) no Neptune ML

As redes neurais para grafos (GNNs) pertencem a uma família de redes neurais que calculam representações de nós levando em consideração a estrutura e os atributos dos nós próximos. As GNNs complementam outros métodos tradicionais de machine learning e rede neural que não são adequados para dados de grafos.

As GNNs são usadas para resolver tarefas de machine learning, como classificação e regressão de nós (previsão de propriedades de nós) e classificação e regressão de bordas (previsão de propriedades de bordas) ou previsão de links (previsão se dois nós no grafo devem estar conectados ou não).

Em geral, o uso de uma GNN para uma tarefa de machine learning envolve duas fases:

- Uma fase de codificação, em que s GNN calcula um vetor d-dimensional para cada nó no grafo. Esses vetores também são denominados representações ou incorporações.
- Uma fase de decodificação, que faz previsões com base nas representações codificadas.

Para classificação e regressão de nós, as representações de nós são usadas diretamente para as tarefas de classificação e regressão. Para classificação e regressão de bordas, as representações dos nós incidentes em uma borda são usadas como entrada para a classificação ou a regressão. Para previsão de links, uma pontuação de probabilidade de borda é calculada usando um par de representações de nós e uma representação do tipo de borda.

A Deep Graph Library (DGL) viabiliza a definição e o treinamento eficientes de GNNs para essas tarefas.

Diferentes modelos de GNN são unificados sob a formulação da transmissão de mensagens. Nessa visualização, a representação de um nó em um grafo é calculada usando as representações dos vizinhos do nó (as mensagens), junto com a representação inicial do nó. No NeptuneML, a representação inicial de um nó é derivada dos atributos extraídos de suas propriedades do nó ou pode ser aprendida e depende da identidade do nó.

O Neptune ML também oferece a opção de concatenar atributos de nós e representações de nós que podem ser aprendidos para servir como representação do nó original.

Para as várias tarefas no Neptune ML envolvendo grafos com propriedades de nós, usamos a Relational Graph Convolutional Network (R-GCN) para realizar a fase de codificação. A R-GCN é uma arquitetura de GNN adequada para grafos com vários tipos de nós e bordas (conhecidos como grafos heterogêneos).

A rede R-GCN consiste em um número fixo de camadas, empilhadas uma após a outra. Cada camada da R-GCN usa seus parâmetros de modelo que podem ser aprendidos para agregar informações da vizinhança imediata de 1 salto de um nó. Como as camadas subsequentes usam as representações de saída da camada anterior como entrada, o raio da vizinhança do grafo que influencia a incorporação final de um nó depende do número de camadas (`num-layer`) da rede R-GCN.

Por exemplo, isso significa que uma rede de duas camadas usa informações de nós que estão a dois saltos de distância.

Para saber mais sobre GNNs, consulte A Comprehensive Survey on Graph Neural Networks. Para obter mais informações sobre a Deep Graph Library (DGL), visite a página da web da DGL. Para ver um tutorial prático sobre como usar a DGL com GNNs, consulte Learning graph neural networks with Deep Graph Library.

Treinar redes neurais para grafos

No machine learning, o processo de obter um modelo para aprender a fazer boas previsões para uma tarefa é chamado de treinamento de modelos. Em geral, isso é realizado especificando um objetivo específico a ser otimizado, bem como um algoritmo a ser usado para realizar essa otimização.

Esse processo também é empregado no treinamento de uma GNN para aprender boas representações para a tarefa posterior. Criamos uma função objetiva para essa tarefa que é minimizada durante o treinamento de modelos. Por exemplo, para classificação de nós, usamos CrossEntropyLoss como o objetivo, o que penaliza erros de classificação, e para regressão de nós minimizamos o MeanSquareError.

O objetivo geralmente é uma função de perda que pega as previsões do modelo para um ponto de dados específico e as compara ao valor verdadeiro básico desse ponto de dados. Ela exibe o valor da perda, que mostra o quão distantes estão as previsões do modelo. O objetivo do processo

de treinamento é minimizar a perda e garantir que as previsões do modelo estejam próximas da verdade.

O algoritmo de otimização usado no machine learning para o processo de treinamento geralmente é uma variante do gradiente descendente. No Neptune ML, usamos Adam, que é um algoritmo para otimização baseada em gradiente de primeira ordem de funções objetivas estocásticas com base em estimativas adaptativas de momentos de ordem inferior.

Embora o processo de treinamento de modelos tente garantir que os parâmetros do modelo aprendido estejam próximos dos mínimos da função objetiva, o desempenho geral de um modelo também depende dos hiperparâmetros do modelo, que são configurações do modelo que não são aprendidas pelo algoritmo de treinamento. Por exemplo, a dimensionalidade da representação do nó aprendido, `num-hidden`, é um hiperparâmetro que afeta o desempenho do modelo. Portanto, é comum no machine learning realizar a otimização de hiperparâmetros (HPO) para escolher os hiperparâmetros adequados.

O Neptune ML usa uma tarefa de ajuste de hiperparâmetros do SageMaker para iniciar várias instâncias de treinamento de modelos com diferentes configurações de hiperparâmetros para tentar encontrar o melhor modelo para uma variedade de configurações de hiperparâmetros. Consulte Personalizar as configurações de hiperparâmetros do modelo no Neptune ML.

Modelos de incorporação de grafos de conhecimento no Neptune ML

Grafos de conhecimento (KGs) são grafos que codificam informações sobre diferentes entidades (nós) e suas relações (bordas). No Neptune ML, os modelos de incorporação de grafos de conhecimento são aplicados por padrão para realizar a previsão de links quando o grafo não contém propriedades de nós, somente relações com outros nós. Embora os modelos de R-GCN com incorporações que podem ser aprendidas também possam ser usados para esses grafos especificando o tipo de modelo como `"rgcn"`, os modelos de incorporação de grafos de conhecimento são mais simples e foram projetados para ser eficazes para aprender representações de grafos de conhecimento em grande escala.

Os modelos de incorporação de grafos de conhecimento são usados em uma tarefa de previsão de links para prever os nós ou relações que completam uma `(h, r, t)` tripla em que **`h`** é o nó de origem, **`r`** é o tipo de relação e **`t`** é o nó de destino.

Os modelos de incorporação de grafos de conhecimento implementados no Neptune ML são `distmult`, `transE` e `rotatE`. Para saber mais sobre os modelos de incorporação de grafos de conhecimento, consulte DGL-KE.

Treinar modelos personalizados no Neptune ML

O Neptune ML permite definir e implementar os próprios modelos personalizados, para cenários específicos. Consulte Modelos personalizados no Neptune ML para obter informações sobre como implementar um modelo personalizado e como usar a infraestrutura do Neptune ML para treiná-lo.

Personalizar as configurações de hiperparâmetros do modelo no Neptune ML

Quando você inicia um trabalho de treinamento de modelos do Neptune ML, o Neptune ML usa automaticamente as informações inferidas do trabalho de processamento de dados anterior. Ele usa as informações para gerar intervalos de configuração de hiperparâmetros que são usados para criar um trabalho de ajuste de hiperparâmetros do SageMaker para treinar vários modelos para sua tarefa. Dessa forma, você não precisa especificar uma longa lista de valores de hiperparâmetros com os quais os modelos serão treinados. Em vez disso, os intervalos e os padrões dos hiperparâmetros do modelo são selecionados com base no tipo de tarefa, no tipo de grafo e nas configurações do trabalho de ajuste.

No entanto, você também pode substituir a configuração padrão de hiperparâmetros e fornecer hiperparâmetros personalizados modificando um arquivo de configuração JSON que o trabalho de processamento de dados gera.

Usando a API modelTraining do Neptune ML, é possível controlar várias configurações de trabalhos de ajuste de hiperparâmetros gerais, como `maxHPONumberOfTrainingJobs`, `maxHPOParallelTrainingJobs` e `trainingInstanceType`. Para um controle mais refinado sobre os hiperparâmetros do modelo, você pode personalizar o arquivo `model-HPO-configuration.json` que o trabalho de processamento de dados gera. O arquivo é salvo no local do Amazon S3 que você especificou para a saída do trabalho de processamento.

É possível baixar o arquivo, editá-lo para substituir as configurações padrão de hiperparâmetros e enviá-lo de volta para o mesmo local do Amazon S3. Não altere o nome do arquivo e siga estas instruções ao editar.

Como baixar o arquivo do Amazon S3:

```
aws s3 cp \
  s3://(bucket name)/(path to output folder)/model-HPO-configuration.json \
  ./
```

Quando terminar de editar, faça upload do arquivo de volta para onde estava:

```
aws s3 cp \
  model-HPO-configuration.json \
  s3://(bucket name)/(path to output folder)/model-HPO-configuration.json
```

Estrutura do arquivo **`model-HPO-configuration.json`**

O arquivo `model-HPO-configuration.json` especifica o modelo a ser treinado, o `task_type` de machine learning e os hiperparâmetros que devem ser variados ou fixos para as várias execuções de treinamento do modelo.

Os hiperparâmetros são categorizados como pertencentes a vários níveis, o que significa a precedência dada aos hiperparâmetros quando o trabalho de ajuste do hiperparâmetro é invocado:

- Os hiperparâmetros de nível 1 têm a maior precedência. Se você definir `maxHPONumberOfTrainingJobs` como um valor menor que 10, somente os hiperparâmetros de nível 1 serão ajustados, e o restante assumirá seus valores padrão.
- Os hiperparâmetros de nível 2 têm menor precedência; portanto, se você tiver mais de 10, mas menos de 50 trabalhos de treinamento no total para um trabalho de ajuste, os hiperparâmetros de nível 1 e nível 2 serão ajustados.
- Os hiperparâmetros de nível 3 são ajustados junto com os níveis 1 e 2 somente se você tiver mais de 50 trabalhos de treinamento no total.
- Por fim, os hiperparâmetros fixos não são ajustados e sempre assumem seus valores padrão.

Exemplo de arquivo **`model-HPO-configuration.json`**

Veja a seguir um exemplo de arquivo `model-HPO-configuration.json`:

```
{
  "models": [
    {
      "model": "rgcn",
      "task_type": "node_class",
      "eval_metric": {
        "metric": "acc"
      },
      "eval_frequency": {
          "type":  "evaluate_every_epoch",
          "value":  1
      },
      "1-tier-param": [
        {
            "param": "num-hidden",
            "range": [16, 128],
            "type": "int",
```

```
            "inc_strategy": "power2"
        },
        {
          "param": "num-epochs",
          "range": [3,30],
          "inc_strategy": "linear",
          "inc_val": 1,
          "type": "int",
          "node_strategy": "perM"
        },
        {
          "param": "lr",
          "range": [0.001,0.01],
          "type": "float",
          "inc_strategy": "log"
        }
      ],
      "2-tier-param": [
        {
          "param": "dropout",
          "range": [0.0,0.5],
          "inc_strategy": "linear",
          "type": "float",
          "default": 0.3
        },
        {
          "param": "layer-norm",
          "type": "bool",
          "default": true
        }
      ],
      "3-tier-param": [
        {
          "param": "batch-size",
          "range": [128, 4096],
          "inc_strategy": "power2",
          "type": "int",
          "default": 1024
        },
        {
          "param": "fanout",
          "type": "int",
          "options": [[10, 30],[15, 30], [15, 30]],
          "default": [10, 15, 15]
```

```
        },
        {
          "param": "num-layer",
          "range": [1, 3],
          "inc_strategy": "linear",
          "inc_val": 1,
          "type": "int",
          "default": 2
        },
        {
          "param": "num-bases",
          "range": [0, 8],
          "inc_strategy": "linear",
          "inc_val": 2,
          "type": "int",
          "default": 0
        }
      ],
      "fixed-param": [
        {
          "param": "concat-node-embed",
          "type": "bool",
          "default": true
        },
        {
          "param": "use-self-loop",
          "type": "bool",
          "default": true
        },
        {
          "param": "low-mem",
          "type": "bool",
          "default": true
        },
        {
          "param": "l2norm",
          "type": "float",
          "default": 0
        }
      ]
    }
  ]
}
```

Elementos de um arquivo **`model-HPO-configuration.json`**

O arquivo contém um objeto JSON com uma única matriz de nível superior chamada `models` que contém um único objeto de configuração de modelo. Ao personalizar o arquivo, assegure-se de que a matriz `models` tenha somente um objeto de configuração de modelo. Se seu arquivo contiver mais de um objeto de configuração de modelo, o trabalho de ajuste falhará com um aviso.

O objeto de configuração do modelo contém os seguintes elementos gerais:

- **`model`**: (string) O tipo de modelo a ser treinado (não modifique). Os valores válidos são:
  - `"rgcn"`: é o padrão para tarefas de classificação e regressão de nós e para tarefas de previsão de links heterogêneos.
  - `"transe"`: é o padrão para as tarefas de previsão de links do KGE.
  - `"distmult"`: é um tipo de modelo alternativo para tarefas de previsão de links do KGE.
  - `"rotate"`: é um tipo de modelo alternativo para tarefas de previsão de links do KGE.

  Como regra, não modifique diretamente o valor `model`, pois tipos de modelos diferentes geralmente têm hiperparâmetros aplicáveis substancialmente diferentes, o que pode gerar um erro de análise após o início do trabalho de treinamento.

  Para alterar o tipo de modelo, use o parâmetro `modelName` na API modelTraining em vez de alterá-lo no arquivo `model-HPO-configuration.json`.

  Uma forma de alterar o tipo de modelo e fazer alterações detalhadas nos hiperparâmetros é copiar o modelo padrão de configuração do modelo no modelo que você deseja usar e colá-lo no arquivo `model-HPO-configuration.json`. Haverá uma pasta denominada `hpo-configuration-templates` no mesmo local do Amazon S3 que o arquivo `model-HPO-configuration.json` se o tipo de tarefa inferido for compatível com vários modelos. Essa pasta contém todas as configurações padrão de hiperparâmetros para os outros modelos aplicáveis à tarefa.

  Por exemplo, se você quiser alterar as configurações de modelo e hiperparâmetros para uma tarefa de previsão de link `KGE` do modelo padrão `transe` para um modelo `distmult`, basta colar o conteúdo do arquivo `hpo-configuration-templates/distmult.json` no arquivo `model-HPO-configuration.json` e editar os hiperparâmetros conforme necessário.

  > **Note**
  >
  > Se você definir o parâmetro `modelName` na API `modelTraining` e também alterar a especificação `model` e de hiperparâmetros no arquivo `model-HPO-`

`configuration.json`, e elas forem diferentes, o valor `model` no arquivo `model-HPO-configuration.json` terá precedência, e o valor `modelName` será ignorado.

- **`task_type`**: (String) O tipo de tarefa de machine learning inferido ou passado diretamente para o trabalho de processamento de dados (não modifique). Os valores válidos são:
  - `"node_class"`
  - `"node_regression"`
  - `"link_prediction"`

  O trabalho de processamento de dados infere o tipo de tarefa examinando o conjunto de dados exportado e o arquivo de configuração do trabalho de treinamento gerado para conferir as propriedades do conjunto de dados.

  Esse valor não deve ser alterado. Se você quiser treinar uma tarefa diferente, precisará executar um novo trabalho de processamento de dados. Se o valor `task_type` não for o esperado, confira as entradas do seu trabalho de processamento de dados para ter certeza de que estão corretas. Isso inclui parâmetros para a API `modelTraining`, bem como no arquivo de configuração do trabalho de treinamento gerado pelo processo de exportação de dados.
- **`eval_metric`**: (String) a métrica de avaliação deve ser usada para avaliar o desempenho do modelo e selecionar o modelo com melhor desempenho em todas as execuções de HPO. Os valores válidos são:
  - `"acc"`: precisão de classificação padrão. É o padrão para tarefas de classificação de rótulo único, a menos que rótulos não equilibrados sejam encontrados durante o processamento de dados. Nesse caso, o padrão será `"F1"`.
  - `"acc_topk"`: o número de vezes que o rótulo correto está entre as principais previsões **k**. Também é possível definir o valor **k** transmitindo `topk` como uma chave extra.
  - `"F1"`: a pontuação da F1.
  - `"mse"`: métrica de erro quadrático médio, para tarefas de regressão.
  - `"mrr"`: métrica média de classificação recíproca.
  - `"precision"`: a precisão do modelo, calculada como a razão entre os positivos verdadeiros e os positivos previstos: `= true-positives / (true-positives + false-positives)`.
  - `"recall"`: a chamada do modelo, calculada como a razão entre os positivos verdadeiros e os positivos reais: `= true-positives / (true-positives + false-negatives)`.
  - `"roc_auc"`: a área abaixo da curva ROC. Esse é o padrão para classificação com vários rótulos.

Por exemplo, para alterar a métrica para `F1`, altere o valor `eval_metric` da seguinte forma:

```
"  eval_metric": {
    "metric": "F1",
  },
```

Ou para alterar a métrica para uma pontuação de precisão `topk`, você mudaria a `eval_metric` da seguinte forma:

```
  "eval_metric": {
    "metric": "acc_topk",
    "topk": 2
  },
```

- **`eval_frequency`**: (Objeto) Especifica com que frequência durante o treinamento o desempenho do modelo no conjunto de validação deve ser conferido. Com base no desempenho da validação, a parada antecipada pode então ser iniciada e o melhor modelo pode ser salvo.

  O objeto `eval_frequency` contém dois elementos, a saber, `"type"` e `"value"`. Por exemplo:

  ```
    "eval_frequency": {
      "type":  "evaluate_every_pct",
      "value":  0.1
    },
  ```

  Os valores `type` válidos são:

  - **`evaluate_every_pct`**: especifica a porcentagem de treinamento a ser concluído para cada avaliação.

    Para `evaluate_every_pct`, o campo `"value"` contém um número de ponto flutuante entre zero e um que expressa essa porcentagem.

  - **`evaluate_every_batch`**: especifica o número de treinamento a ser concluído para cada avaliação.

    Para `evaluate_every_batch`, o campo `"value"` contém um número inteiro que expressa essa contagem de lotes.

- **`evaluate_every_epoch`**: especifica o número de épocas por avaliação, em que uma nova época começa à meia-noite.

  Para `evaluate_every_epoch`, o campo `"value"` contém um número inteiro que expressa essa contagem de épocas.

A configuração padrão para `eval_frequency` é:

```
"eval_frequency": {
  "type":  "evaluate_every_epoch",
  "value":  1
},
```

- **`1-tier-param`**: (Obrigatório) Uma matriz de hiperparâmetros de nível 1.

  Se não quiser ajustar nenhum hiperparâmetro, você poderá definir isso como uma matriz vazia. Isso não afeta o número total de trabalhos de treinamento lançados pelo trabalho de ajuste de hiperparâmetros do SageMaker. Significa apenas que todos os trabalhos de treinamento, se houver mais de 1, mas menos de 10, serão executados com o mesmo conjunto de hiperparâmetros.

  Por outro lado, se você quiser tratar todos os seus hiperparâmetros ajustáveis com igual importância, poderá colocar todos os hiperparâmetros nessa matriz.
- **`2-tier-param`**: (obrigatório) uma matriz de hiperparâmetros de nível 2.

  Esses parâmetros serão ajustados somente se `maxHPONumberOfTrainingJobs` tiver um valor maior que 10. Caso contrário, eles serão fixados nos valores padrão.

  Se você tiver um orçamento de treinamento de no máximo 10 trabalhos de treinamento ou não quiser hiperparâmetros de nível 2 por qualquer outro motivo, mas quiser ajustar todos os hiperparâmetros ajustáveis, defina isso como uma matriz vazia.
- **`3-tier-param`**: (obrigatório) uma matriz de hiperparâmetros de nível 3.

  Esses parâmetros serão ajustados somente se `maxHPONumberOfTrainingJobs` tiver um valor maior que 50. Caso contrário, eles serão fixados nos valores padrão.

  Se não quiser ajustar nenhum hiperparâmetro de nível 3, você poderá definir isso como uma matriz vazia.

- **`fixed-param`**: (obrigatório) uma matriz de hiperparâmetros fixos que usam apenas seus valores padrão e não variam em diferentes trabalhos de treinamento.

  Se quiser variar todos os hiperparâmetros, você poderá definir isso como uma matriz vazia e definir o valor como `maxHPONumberOfTrainingJobs` grande o suficiente para variar todos os níveis ou tornar todos os hiperparâmetros de nível 1.

O objeto JSON que representa cada hiperparâmetro em `1-tier-param`, `2-tier-param`, `3-tier-param` e `fixed-param` contém os seguintes elementos:

- **`param`**: (string) o nome do hiperparâmetro (não altere).

  Veja a lista de nomes de hiperparâmetros válidos no Neptune ML.
- **`type`**: (string) o tipo de hiperparâmetro (não altere).

  Os tipos válidos são `bool`, `int` e `float`.
- **`default`**: (string) o valor padrão para o hiperparâmetro.

  Você pode definir um novo valor padrão.

Os hiperparâmetros ajustáveis também podem conter os seguintes elementos:

- **`range`**: (matriz) o intervalo de um hiperparâmetro ajustável contínuo.

  Deve ser uma matriz com dois valores, ou seja, o mínimo e o máximo do intervalo (`[min, max]`).
- **`options`**: (matriz) as opções para um hiperparâmetro categórico ajustável.

  Essa matriz deve conter todas as opções a serem consideradas:

  ```
  "options" : [value1, value2, ... valuen]
  ```
- **`inc_strategy`**: (string) o tipo de alteração incremental para intervalos contínuos de hiperparâmetros ajustáveis (não altere).

  Os valores válidos são `log`, `linear` e `power2`. Aplica-se somente quando a chave de intervalo é definida.

  Modificar isso pode resultar no não uso de todo o intervalo de seu hiperparâmetro para ajuste.

- **`inc_val`** (float) a quantidade pela qual os incrementos sucessivos diferem para hiperparâmetros ajustáveis contínuos (não altere).

  Aplica-se somente quando a chave de intervalo é definida.

  Modificar isso pode resultar no não uso de todo o intervalo de seu hiperparâmetro para ajuste.
- **`node_strategy`**: (string) indica que o intervalo efetivo desse hiperparâmetro deve mudar com base no número de nós no grafo (não altere).

  Os valores válidos são `"perM"` (por milhão), `"per10M"` (por 10 milhões) e `"per100M"` (por 100 milhões).

  Em vez de alterar esse valor, altere `range`.
- **`edge_strategy`**: (string) indica que o intervalo efetivo desse hiperparâmetro deve mudar com base no número de bordas no grafo (não altere).

  Os valores válidos são `"perM"` (por milhão), `"per10M"` (por 10 milhões) e `"per100M"` (por 100 milhões).

  Em vez de alterar esse valor, altere `range`.

Lista de todos os hiperparâmetros no Neptune ML

A lista a seguir contém todos os hiperparâmetros que podem ser definidos em qualquer lugar no Neptune ML, para qualquer tipo de modelo e tarefa. Como nem todos são aplicáveis a todos os tipos de modelo, é importante definir apenas hiperparâmetros no arquivo `model-HPO-configuration.json` que aparece para o modelo que você está usando.

- **`batch-size`**: o tamanho do lote de nós-alvo usados em uma passagem direta. Tipo: `int`.

  Definir isso como um valor muito maior pode causar problemas de memória para treinamento em instâncias de GPU.
- **`concat-node-embed`**: indica se é necessário obter a representação inicial de um nó concatenando seus atributos processados com incorporações de nós iniciais que podem ser aprendidos a fim de aumentar a expressividade do modelo. Tipo: `bool`.
- **`dropout`**: a probabilidade de abandono aplicada às camadas de abandono. Tipo: `float`.

- **`edge-num-hidden`**: o tamanho da camada oculta ou o número de unidades do módulo de atributos de borda. Usado somente quando `use-edge-features` for definido como `True`. Tipo: flutuante.
- **`enable-early-stop`**: alterna se deve ou não usar o atributo de parada antecipada. Tipo: `bool`. Padrão: `true`.

  Use esse parâmetro booliano para desativar o atributo de parada antecipada.
- **`fanout`**: o número de vizinhos para amostra para um nó-alvo durante a amostragem de vizinhos. Tipo: `int`.

  Esse valor está fortemente acoplado a `num-layers` e deve estar sempre no mesmo nível de hiperparâmetros. Isso ocorre porque você pode especificar um fanout para cada camada potencial do GNN.

  Como esse hiperparâmetro pode fazer com que o desempenho do modelo varie amplamente, ele deve ser corrigido ou definido como um hiperparâmetro de nível 2 ou nível 3. Definir isso como um valor grande pode causar problemas de memória para treinamento em instâncias de GPU.
- **`gamma`**: o valor da margem na função de pontuação. Tipo: `float`.

  Isso se aplica somente aos modelos de previsão de links `KGE`.
- **`l2norm`**: o valor de redução de peso usado no otimizador que impõe uma penalidade de normalização L2 nos pesos. Tipo: `bool`.
- **`layer-norm`**: indica se a normalização de camadas deve ser usada para modelos `rgcn`. Tipo: `bool`.
- **`low-mem`**: indica se deve ser usada uma implementação de baixa memória da função de transmissão de mensagens de relação em detrimento da velocidade. Tipo: `bool`.
- **`lr`**: a taxa de aprendizado. Tipo: `float`.

  Isso deve ser definido como um hiperparâmetro de nível 1.
- **`neg-share`**: na previsão de links, indica se as amostras de bordas positivas podem compartilhar amostras de bordas negativas. Tipo: `bool`.
- **`num-bases`**: o número de bases para decomposição de bases em um modelo `rgcn`. Usar um valor de `num-bases` menor do que o número de tipos de borda no grafo atua como um regularizador para o modelo `rgcn`. Tipo: `int`.
- **`num-epochs`**: o número de épocas de treinamento a serem executados. Tipo: `int`.

Época é uma passagem de treinamento completa pelo grafo.

- **num-hidden**: o tamanho da camada oculta ou o número de unidades. Tipo: `int`.

  Isso também define o tamanho inicial de incorporação para nós sem atributos.

  Definir isso como um valor muito maior sem reduzir `batch-size` pode causar problemas de falta de memória para treinamento em instâncias de GPU.

- **num-layer**: o número de camadas de GNN no modelo. Tipo: `int`.

  Esse valor está fortemente acoplado ao parâmetro fanout e deve vir depois que o fanout é definido no mesmo nível de hiperparâmetros.

  Como isso ode fazer com que o desempenho do modelo varie amplamente, ele deve ser corrigido ou definido como um hiperparâmetro de nível 2 ou nível 3.

- **num-negs**: na previsão de links, o número de amostras negativas por amostra positiva. Tipo: `int`.

- **per-feat-name-embed**: indica se cada atributo deve ser incorporado transformando-o de forma independente antes de combinar os atributos. Tipo: `bool`.

  Quando definido como `true`, cada atributo por nó é transformado de forma independente em um tamanho de dimensão fixo antes que todos os atributos transformados do nó sejam concatenados e posteriormente transformados na dimensão `num_hidden`.

  Quando definido como `false`, os atributos são concatenados sem nenhuma transformação específica do atributo.

- **regularization-coef**: na previsão de links, o coeficiente de perda de regularização. Tipo: `float`.

- **rel-part**: indica se a partição de relação deve ser usada para previsão de links `KGE`. Tipo: `bool`.

- **sparse-lr**: a taxa de aprendizado para incorporações de nós que podem ser aprendidos. Tipo: `float`.

  As incorporações de nós iniciais que podem ser aprendidos são usadas para nós sem atributos ou quando `concat-node-embed` está definido. Os parâmetros da camada de incorporação de nós esparsos que podem ser aprendidos são treinados usando um otimizador separado que pode ter uma taxa de aprendizado separada.

- **`use-class-weight`**: indica se devem ser aplicados pesos de classe para tarefas de classificação não equilibradas. Se definido como `true`, as contagens de rótulos são usadas para definir um peso para cada rótulo de classe. Tipo: `bool`.
- **`use-edge-features`**: indica se os atributos de borda devem ser usados durante a transmissão de mensagens. Se definido como `true`, um módulo de atributo de borda personalizado é adicionado à camada RGCN para tipos de borda que têm atributos. Tipo: `bool`.
- **`use-self-loop`**: indica se deve incluir loops automáticos no treinamento de um modelo `rgcn`. Tipo: `bool`.
- **`window-for-early-stop`**: controla a média do número das pontuações de validação mais recentes para decidir sobre uma parada antecipada. O padrão é 3. type=int. Consulte também Interrupção antecipada do processo de treinamento de modelos no Neptune ML. Tipo: `int`. Padrão: `3`.

  Consulte .

Personalizar hiperparâmetros no Neptune ML

Ao editar o arquivo `model-HPO-configuration.json`, os seguintes tipos de alteração a serem feitas são os mais comuns:

- Edite os valores mínimo e/ou máximo de hiperparâmetros `range`.
- Defina um hiperparâmetro como um valor fixo movendo-o para a seção `fixed-param` e definindo seu valor padrão para o valor fixo que você deseja que ele assuma.
- Altere a prioridade de um hiperparâmetro colocando-o em um nível específico, editando seu intervalo e certificando-se de que seu valor padrão seja definido adequadamente.

Práticas recomendadas para treinamento de modelos

Há procedimentos que você pode realizar para melhorar o desempenho dos modelos do Neptune ML.

Escolher a propriedade correta do nó

Pode ser que nem todas as propriedades no grafo sejam significativas ou relevantes para as tarefas de machine learning. Todas as propriedades irrelevantes devem ser excluídas durante a exportação de dados.

Veja algumas práticas recomendadas:

- Use especialistas no domínio para ajudar a avaliar a importância dos atributos e a viabilidade de usá-los para previsões.
- Remova os atributos considerados redundantes ou irrelevantes para reduzir o ruído nos dados e as correlações sem importância.
- Faça iterações à medida que constrói o modelo. Ajuste os atributos, as combinações de atributos e os objetivos de ajuste à medida que avança.

O processamento de atributos no Guia do desenvolvedor do Amazon Machine Learning apresenta diretrizes adicionais para o processamento de atributos relevantes para o Neptune ML.

Lidar com pontos de dados atípicos

Valor atípico é um ponto de dados significativamente diferente dos dados restantes. Valores atípicos podem prejudicar ou induzir erros no processo de treinamento, ocasionando maior tempo de treinamento ou modelos menos precisos. A menos que sejam realmente importantes, você deve eliminar os valores atípicos antes de exportar os dados.

Remover bordas e nós duplicados

Os grafos armazenados no Neptune podem ter bordas ou nós duplicados. Esses elementos redundantes introduzirão ruído no treinamento de modelos de ML. Elimine bordas ou nós duplicados antes de exportar os dados.

Ajustar a estrutura do grafo

Quando o grafo é exportado, você pode alterar a forma como os atributos são processados e como o grafo é construído, para melhorar o desempenho do modelo.

Veja algumas práticas recomendadas:

- Quando uma propriedade de borda tem o significado de categorias de bordas, em alguns casos vale a pena transformá-la em tipos de borda.
- A política de normalização padrão usada para uma propriedade numérica é `min-max`, mas em alguns casos outras políticas de normalização funcionam melhor. É possível pré-processar a propriedade e alterar a política de normalização conforme explicado em Elementos de um arquivo `model-HPO-configuration.json`.
- O processo de exportação gera automaticamente tipos de atributo com base nos tipos de propriedade. Por exemplo, ele trata as propriedades `String` como atributos categóricos e as propriedades `Float` e `Int` como atributos numéricos. Se necessário, você poderá modificar o tipo de atributo após a exportação (consulte Elementos de um arquivo `model-HPO-configuration.json`).

Ajustar os intervalos e padrões dos hiperparâmetros

A operação de processamento de dados infere intervalos de configuração de hiperparâmetros a partir do grafo. Se os intervalos e os padrões de hiperparâmetros do modelo gerado não funcionarem bem para os dados dos grafos, você poderá editar o arquivo de configuração do HPO para criar sua própria estratégia de ajuste de hiperparâmetros.

Veja algumas práticas recomendadas:

- Quando o grafo fica grande, o tamanho padrão da dimensão oculta pode não ser grande o suficiente para conter todas as informações. É possível alterar o hiperparâmetro `num-hidden` para controlar o tamanho da dimensão oculta.
- Para modelos de incorporação de grafos de conhecimento (KGE), convém alterar o modelo específico que está sendo usado de acordo com a estrutura do grafo e o orçamento.

  Os modelos `TrainsE` têm dificuldade de lidar com relações um-para-muitos (1-N), muitos-para-um (N-1) e muitos-para-muitos (N-N). Os modelos `DistMult` têm dificuldade de lidar com relações simétricas. `RotatE` é bom em modelar todos os tipos de relações, mas é mais caro do que `TrainsE` e `DistMult` durante o treinamento.
- Em alguns casos, quando tanto a identificação do nó quanto as informações do atributo do nó são importantes, você deve usar `` `concat-node-embed` `` para instruir o modelo Neptune ML a obter a representação inicial de um nó concatenando os atributos com suas incorporações iniciais.

- Ao obter um desempenho razoavelmente bom em alguns hiperparâmetros, é possível ajustar o espaço de pesquisa de hiperparâmetros de acordo com esses resultados.

Interrupção antecipada do processo de treinamento de modelos no Neptune ML

A interrupção antecipada pode reduzir significativamente o tempo de execução do treinamento de modelos e os custos associados sem degradar o desempenho do modelo. Também evita que o modelo se ajuste demais aos dados de treinamento.

A interrupção antecipada depende de medições regulares do desempenho do conjunto de validação. Inicialmente, o desempenho melhora à medida que o treinamento prossegue, mas quando o modelo começa a se ajustar demais, ele começa a declinar novamente. O atributo de interrupção antecipada identifica o ponto em que o modelo começa a se ajustar demais e interrompe o treinamento do modelo nesse ponto.

O Neptune ML monitora as chamadas da métrica de validação e compara a métrica de validação mais recente com a média das métricas de validação nas últimas **n** avaliações, em que **n** é um número definido usando o parâmetro `window-for-early-stop`. Assim que a métrica de validação for pior do que a média, o Neptune ML interromperá o treinamento do modelo e salvará o melhor modelo até o momento.

É possível controlar a interrupção antecipada usando os seguintes parâmetros:

- **`window-for-early-stop`**: o valor desse parâmetro é um número inteiro que especifica a média do número de pontuações de validação recentes ao decidir sobre uma interrupção antecipada. O valor padrão é `3`.
- **`enable-early-stop`**: use esse parâmetro booliano para desativar o atributo de interrupção antecipada. Por padrão, o valor é `true`.

Interrupção antecipada do processo HPO no Neptune ML

O atributo de interrupção antecipada no Neptune ML também interrompe os trabalhos de treinamento que não estão funcionando bem em comparação com outros trabalhos de treinamento, usando o atributo de início a quente do HPO do SageMaker. Isso também pode reduzir custos e melhorar a qualidade do HPO.

Consulte Run a warm start hyperparameter tuning job para obter uma descrição de como isso funciona.

O início a quente oferece a capacidade de transmitir informações aprendidas em trabalhos de treinamento anteriores para trabalhos de treinamento subsequentes e oferece dois benefícios distintos:

- Primeiro, os resultados dos trabalhos de treinamento anteriores são usados para selecionar boas combinações de hiperparâmetros a serem pesquisados no novo trabalho de ajuste.
- Segundo, permite a interrupção antecipada para acessar mais execuções do modelo, o que reduz o tempo de ajuste.

Esse atributo é habilitado automaticamente no Neptune ML e permite que você encontre um equilíbrio entre o tempo de treinamento e o desempenho do modelo. Se você estiver satisfeito com o desempenho do modelo atual, poderá usar esse modelo. Caso contrário, execute mais HPOs que são iniciados a quente com os resultados das execuções anteriores para descobrir um modelo melhor.

Obter serviços de suporte profissional

A AWS oferece serviços de suporte profissional para ajudar você com problemas em machine learning em projetos do Neptune. Se tiver problemas, entre em contato com o AWS Support.

Usar um modelo treinado para gerar novos artefatos de modelo

Usando o comando de transformação de modelos do Neptune ML, é possível calcular artefatos de modelo, como incorporações de nós em dados de grafos processados, usando parâmetros de modelo pré-treinados.

Transformação de modelos para inferência incremental

No fluxo de trabalho de inferência de modelos incrementais, depois de processar os dados de grafos atualizados que você exportou do Neptune, é possível iniciar um trabalho de processamento de modelos usando um comando do curl (ou awscurl) como o seguinte:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltransform
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "mlModelTrainingJobId": "(the ML model training job-id)",
        "modelTransformOutputS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-
transform/"
      }'
```

Depois, é possível transmitir o ID desse trabalho para a chamada da API create-endpoints a fim de criar um endpoint ou atualizar um existente com os novos artefatos do modelo gerados por esse trabalho. Isso permite que o endpoint novo ou atualizado forneça previsões de modelo para os dados de grafos atualizados.

Transformação de modelos para qualquer trabalho de treinamento

Você também pode fornecer um parâmetro `trainingJobName` para gerar artefatos de modelo para qualquer um dos trabalhos de treinamento do SageMaker iniciados durante o treinamento de modelos do Neptune ML. Como um trabalho de treinamento de modelos do Neptune ML pode iniciar muitos trabalhos de treinamento do SageMaker, isso oferece a flexibilidade de criar um endpoint de inferência com base em qualquer um desses trabalhos de treinamento do SageMaker.

Por exemplo:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltransform
```

```
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "trainingJobName" : "(name a completed SageMaker training job)",
        "modelTransformOutputS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-
transform/"
      }'
```

Se o trabalho de treinamento original foi para um modelo personalizado fornecido pelo usuário, você deverá incluir um objeto `customModelTransformParameters` ao invocar uma transformação de modelos. Consulte Modelos personalizados no Neptune ML para obter informações sobre como implementar e usar um modelo personalizado.

> **Note**
>
> O comando `modeltransform` sempre executa a transformação de modelos no melhor trabalho de treinamento do SageMaker para esse treinamento.

Consulte O comando modeltransform para obter mais informações sobre trabalhos de treinamento de modelos.

Artefatos produzidos pelo treinamento de modelos no Neptune ML

Após o treinamento de modelos, o Neptune ML usa os parâmetros de modelo mais bem treinados para gerar artefatos de modelo que são necessários para iniciar o endpoint de inferência e fornecer previsões do modelo. Esses artefatos são agrupados pelo trabalho de treinamento e armazenados no local de saída do Amazon S3 do melhor trabalho de treinamento do SageMaker.

As seções a seguir descrevem o que está incluído nos artefatos do modelo para as várias tarefas e como o comando de transformação de modelos usa um modelo treinado preexistente para gerar artefatos mesmo em novos dados de grafos.

Artefatos gerados para diferentes tarefas

O conteúdo dos artefatos do modelo gerados pelo processo de treinamento depende da tarefa de machine learning de destino:

- Classificação e regressão de nós: para previsão de propriedades de nós, os artefatos incluem parâmetros do modelo, incorporações de nós do codificador de GNN, previsões de modelo para nós no grafo de treinamento e alguns arquivos de configuração para o endpoint de inferência. Nas tarefas de classificação e regressão de nós, as previsões do modelo são pré-calculadas para os nós presentes durante o treinamento com o objetivo de reduzir a latência da consulta.
- Classificação e regressão de bordas: para a previsão de propriedades da borda, os artefatos também incluem parâmetros do modelo e incorporações de nós. Os parâmetros do decodificador de modelos são especialmente importantes para inferência porque calculamos as previsões de classificação ou regressão de bordas aplicando o decodificador de modelos às incorporações do vértice de origem e de destino de uma borda.
- Previsão de links: para previsão de links, além dos artefatos gerados para a previsão de propriedades de borda, o grafo de DGL também é incluído como um artefato porque a previsão de links requer o grafo de treinamento para realizar previsões. O objetivo da previsão de links é prever os vértices de destino que provavelmente se combinarão com um vértice de origem para formar uma borda de um tipo específico no grafo. Para fazer isso, a incorporação de nós do vértice de origem e uma representação aprendida para o tipo de borda são combinadas com as incorporações de nós de todos os vértices de destino possíveis a fim de produzir uma pontuação de probabilidade de borda para cada um dos vértices de destino. As pontuações são então classificadas para avaliar os possíveis vértices de destino e gerar os principais candidatos.

Para cada um dos tipos de tarefa, os pesos do modelo de rede neural para grafos da DGL são salvos no artefato do modelo. Isso permite que o Neptune ML calcule novas saídas do modelo à medida que o grafo muda (inferência indutiva), além de usar previsões e incorporações pré-calculadas (inferência transdutiva) para reduzir a latência.

Gerar novos artefatos do modelo

Os artefatos do modelo gerados após o treinamento de modelos no Neptune ML estão diretamente vinculados ao processo de treinamento. Isso significa que as incorporações e as previsões pré-calculadas existem apenas para entidades que estavam no grafo de treinamento original. Embora o modo de inferência indutiva para endpoints do Neptune ML possa calcular previsões para novas entidades em tempo real, convém gerar previsões em lote em novas entidades sem consultar um endpoint.

Para obter previsões do modelo em lote para novas entidades adicionadas ao grafo, novos artefatos do modelo precisam ser recalculados para os novos dados de grafos. Isso é feito com o comando `modeltransform`. Use o comando `modeltransform` quando deseja apenas previsões em lote sem configurar um endpoint ou quando deseja que todas as previsões sejam geradas para que você possa gravá-las de volta no grafo.

Como o treinamento de modelos executa implicitamente uma transformação de modelos no final do processo de treinamento, os artefatos do modelo são sempre recalculados nos dados de grafos de treinamento por um trabalho de treinamento. No entanto, o comando `modeltransform` também pode calcular artefatos de modelo em dados de grafos que não foram usados para treinar um modelo. Para isso, os novos dados de grafos devem ser processados usando as mesmas codificações de atributos dos dados de grafos originais e devem seguir o mesmo esquema de grafo.

É possível fazer isso criando primeiro uma tarefa de processamento de dados que seja clone da tarefa de processamento de dados executada nos dados do grafo de treinamento original e executando-a nos novos dados de grafos (consulte Processar dados de grafos atualizados para o Neptune ML). Depois, chame o comando `modeltransform` com o novo `dataProcessingJobId` e o antigo `modelTrainingJobId` para recalcular os artefatos do modelo nos dados de grafos atualizados.

Para a previsão da propriedade do nó, as incorporações e as previsões do nó são recalculadas nos novos dados de grafos, mesmo para os nós que estavam presentes no grafo de treinamento original.

Para previsão de propriedades de borda e de links, as incorporações de nós também são recalculadas e, da mesma forma, substituem todas as incorporações de nós existentes. Para

recalcular as incorporações de nós, o Neptune ML aplica o codificador de GNN aprendido do modelo treinado anterior aos nós dos novos dados de grafos com seus novos atributos.

Para nós sem atributos, as representações iniciais aprendidas do treinamento do modelo original são reutilizadas. Para novos nós sem atributos e que não estavam presentes no grafo de treinamento original, o Neptune ML inicializa a representação como a média das representações iniciais aprendidas desse tipo de nó presentes no grafo de treinamento original. Isso poderá causar uma certa degradação do desempenho nas previsões do modelo se você tiver muitos nós novos sem atributos, pois todos eles serão inicializados com a incorporação inicial média desse tipo de nó.

Se o modelo for treinado com `concat-node-embed` definido como verdadeiro, as representações iniciais do nó serão criadas concatenando os atributos do nó com a representação inicial que pode ser aprendida. Assim, para o grafo atualizado, a representação inicial dos novos nós também usa a média das incorporações iniciais dos nós, concatenadas com os novos atributos dos nós.

Além disso, no momento, exclusões de nós não são compatíveis. Se os nós tiverem sido removidos no grafo atualizado, você precisará treinar novamente o modelo nos dados do grafo atualizado.

O novo cálculo dos artefatos do modelo reutiliza os parâmetros aprendidos do modelo em um novo grafo e só deve ser feito quando o novo grafo é muito semelhante ao grafo antigo. Se o novo grafo não for suficientemente semelhante, você precisará treinar novamente o modelo para obter um desempenho semelhante nos novos dados de grafos. O que constitui algo suficientemente semelhante depende da estrutura dos dados de grafos, mas, como regra, será necessário treinar novamente o modelo se os novos dados forem mais de 10 a 20% diferentes dos dados do grafo de treinamento original.

Para grafos em que todos os nós têm atributos, a extremidade superior do limite (20% diferente) se aplica, mas para grafos em que muitos nós não têm atributos e os novos nós adicionados ao grafo não têm propriedades, a extremidade inferior (10% diferente) pode até ser alta demais.

Consulte O comando modeltransform para obter mais informações sobre trabalhos de treinamento de modelos.

Modelos personalizados no Neptune ML

O Neptune ML permite definir as próprias implementações de modelos personalizados usando Python. É possível treinar e implantar modelos personalizados usando a infraestrutura do Neptune ML da mesma forma que é feito com os modelos integrados, e usá-los para obter previsões por meio de consultas de grafos.

> **Note**
>
> No momento, a inferência indutiva em tempo real não é compatível com modelos personalizados.

Você pode começar a implementar um modelo personalizado em Python seguindo os exemplos do kit de ferramentas do Neptune ML e usando os componentes do modelo fornecidos no kit de ferramentas do Neptune ML. As seções a seguir fornecem mais detalhes.

Sumário

Visão geral dos modelos personalizados no Neptune ML

Quando usar um modelo personalizado no Neptune ML

Os modelos integrados do Neptune ML lidam com todas as tarefas padrão compatíveis com o Neptune ML, mas pode haver casos em que você queira ter um controle mais granular sobre o modelo para uma tarefa específica ou precise personalizar o processo de treinamento de modelos. Por exemplo, um modelo personalizado é apropriado nas seguintes situações:

- Codificação de atributos de texto de modelos de texto muito grandes precisam ser executados na GPU.
- Você deseja usar seu próprio modelo personalizado de rede neural para grafos (GNN) desenvolvido na Deep Graph Library (DGL).
- Você deseja usar modelos tabulares ou modelos de conjunto para classificação e regressão de nós.

Fluxo de trabalho para desenvolver e usar um modelo personalizado no Neptune ML

O suporte a modelos personalizados no Neptune ML foi projetado para se integrar perfeitamente aos fluxos de trabalho existentes do Neptune ML. Ele funciona executando código personalizado no módulo de origem na infraestrutura do Neptune ML para treinar o modelo. Assim como no caso de um modo integrado, o Neptune ML inicia automaticamente um trabalho de ajuste de hiperparâmetros do SageMaker e seleciona o melhor modelo de acordo com a métrica de avaliação. Depois, ele usa a implementação fornecida no módulo de origem para gerar artefatos de modelo para implantação.

A exportação de dados, a configuração de treinamento e o pré-processamento de dados são os mesmos para modelos personalizados e incorporados.

Depois do pré-processamento de dados, é quando você pode desenvolver e testar de forma iterativa e interativa a implementação de modelos personalizados usando Python. Quando o modelo estiver pronto para produção, você poderá fazer o upload do módulo Python resultante para o Amazon S3 da seguinte forma:

```
aws s3 cp --recursive (source path to module) s3://(bucket name)/(destination path for
 your module)
```

Depois, é possível usar o fluxo de trabalho de dados padrão normal ou incremental para implantar o modelo na produção, com algumas diferenças.

Para treinamento de modelos usando um modelo personalizado, é necessário fornecer um objeto `customModelTrainingParameters` JSON à API de treinamento de modelos do Neptune ML para garantir que o código personalizado seja usado. Os campos no objeto `customModelTrainingParameters` são os seguintes:

- **`sourceS3DirectoryPath`**: (obrigatório) o caminho para o local do Amazon S3 onde o módulo Python que implementa seu modelo está localizado. Isso deve apontar para uma localização válida existente do Amazon S3 que contenha, no mínimo, um script de treinamento, um script de transformação e um arquivo `model-hpo-configuration.json`.
- **`trainingEntryPointScript`**: (opcional) o nome do ponto de entrada no módulo de um script que executa o treinamento de modelos e usa hiperparâmetros como argumentos de linha de comando, incluindo hiperparâmetros fixos.

  Padrão: `training.py`.
- **`transformEntryPointScript`**: (opcional) o nome do ponto de entrada no módulo de um script que deve ser executado após a identificação do melhor modelo da pesquisa de hiperparâmetros, para calcular os artefatos do modelo necessários para a implantação do modelo. Ele deve ser capaz de ser executado sem argumentos de linha de comando.

  Padrão: `transform.py`.

Por exemplo:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltraining
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer"
        "modelName": "custom",
        "customModelTrainingParameters" : {
          "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your Python
 module)",
          "trainingEntryPointScript": "(your training script entry-point name in the
 Python module)",
          "transformEntryPointScript": "(your transform script entry-point name in the
 Python module)"
        }
```

```
      }'
```

Da mesma forma, para habilitar uma transformação de modelos personalizados, você deve fornecer um objeto `customModelTransformParameters` JSON à API de transformação de modelos do Neptune ML, com valores de campo compatíveis com os parâmetros do modelo salvos do trabalho de treinamento. O objeto `customModelTransformParameters` contém os seguintes campos:

- **`sourceS3DirectoryPath`**: (obrigatório) o caminho para o local do Amazon S3 onde o módulo Python que implementa seu modelo está localizado. Isso deve apontar para uma localização válida existente do Amazon S3 que contenha, no mínimo, um script de treinamento, um script de transformação e um arquivo `model-hpo-configuration.json`.
- **`transformEntryPointScript`**: (opcional) o nome do ponto de entrada no módulo de um script que deve ser executado após a identificação do melhor modelo da pesquisa de hiperparâmetros, para calcular os artefatos do modelo necessários para a implantação do modelo. Ele deve ser capaz de ser executado sem argumentos de linha de comando.

  Padrão: `transform.py`.

Por exemplo:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltransform
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "trainingJobName" : "(name of a completed SageMaker training job)",
        "modelTransformOutputS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-
transform/"
        "customModelTransformParameters" : {
          "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your Python
 module)",
          "transformEntryPointScript": "(your transform script entry-point name in the
 Python module)"
        }
      }'
```

Desenvolvimento de modelos personalizados no Neptune ML

Uma boa maneira de começar o desenvolvimento de modelos personalizados é seguir os exemplos do kit de ferramentas do Neptune ML para estruturar e escrever o módulo de treinamento. O kit de ferramentas do Neptune ML também implementa componentes do modelo de ML de grafo modularizado no modelzoo que você pode empilhar e usar para criar o modelo personalizado.

Além disso, o kit de ferramentas fornece funções de utilitário que ajudam a gerar os artefatos necessários durante o treinamento e a transformação de modelos. É possível importar esse pacote Python na implementação personalizada. Todas as funções ou os módulos fornecidos no kit de ferramentas também estão disponíveis no ambiente de treinamento do Neptune ML.

Se o módulo Python tiver dependências externas adicionais, você poderá incluir essas dependências adicionais criando um arquivo `requirements.txt` no diretório do módulo. Os pacotes listados no arquivo `requirements.txt` serão então instalados antes da execução do script de treinamento.

No mínimo, o módulo Python que implementa o modelo personalizado precisa conter o seguinte:

- Um ponto de entrada do script de treinamento
- Um ponto de entrada do script de transformação
- Um arquivo `model-hpo-configuration.json`

Desenvolvimento de scripts de treinamento de modelos personalizados no Neptune ML

O script de treinamento de modelos personalizados deve ser um script Python executável, como o exemplo `train.py` do kit de ferramentas do Neptune ML. Ele deve aceitar nomes e valores de hiperparâmetros como argumentos de linha de comando. Durante o treinamento de modelos, os nomes dos hiperparâmetros são obtidos do arquivo `model-hpo-configuration.json`. Os valores dos hiperparâmetros estarão dentro do intervalo de hiperparâmetros válido se o hiperparâmetro for ajustável ou assumirão o valor padrão do hiperparâmetro se não for ajustável.

O script de treinamento é executado em uma instância de treinamento do SageMaker usando uma sintaxe como esta:

```
python3 (script entry point) --(1st parameter) (1st value) --(2nd parameter) (2nd
 value) (...)
```

Para todas as tarefas, o Neptune ML AutoTrainer envia vários parâmetros necessários ao script de treinamento, além dos hiperparâmetros especificados, e o script deve ser capaz de lidar com esses parâmetros adicionais para funcionar adequadamente.

Esses parâmetros adicionais necessários variam um pouco de acordo com a tarefa:

Para classificação ou regressão de nós

- **`task`**: o tipo de tarefa usado internamente pelo Neptune ML. Para classificação de nós, é `node_class`, e para regressão de nós, é `node_regression`.
- **`model`**: o nome do modelo usado internamente pelo Neptune ML, que é `custom` neste caso.
- **`name`**: o nome da tarefa usada internamente pelo Neptune ML, que é `node_class-custom` para classificação de nós nesse caso, e `node_regression-custom` para regressão de nós.
- **`target_ntype`**: o nome do tipo de nó para classificação ou regressão.
- **`property`**: o nome da propriedade de nó para classificação ou regressão.

Para previsão de links

- **`task`**: o tipo de tarefa usado internamente pelo Neptune ML. Para previsão de links, é `link_predict`.
- **`model`**: o nome do modelo usado internamente pelo Neptune ML, que é `custom` neste caso.
- **`name`**: o nome de tarefa usado internamente pelo Neptune ML, que é `link_predict-custom` nesse caso.

Para classificação ou regressão de bordas

- **`task`**: o tipo de tarefa usado internamente pelo Neptune ML. Para classificação de bordas, é `edge_class`, e para regressão de bordas, é `edge_regression`.
- **`model`**: o nome do modelo usado internamente pelo Neptune ML, que é `custom` neste caso.
- **`name`**: o nome da tarefa usada internamente pelo Neptune ML, que é `edge_class-custom` para classificação de bordas nesse caso, e `edge_regression-custom` para regressão de bordas.
- **`target_etype`**: o nome do tipo de borda para classificação ou regressão.
- **`property`**: o nome da propriedade de borda para classificação ou regressão.

O script deve salvar os parâmetros do modelo, bem como quaisquer outros artefatos necessários ao final do treinamento.

É possível usar as funções do utilitário do kit de ferramentas do Neptune ML para determinar a localização dos dados de grafos processados, o local em que os parâmetros do modelo devem ser salvos e quais dispositivos de GPU estão disponíveis na instância de treinamento. Consulte o exemplo de script de treinamento train.py para ver exemplos de como usar essas funções de utilitário.

Desenvolvimento de scripts de transformação de modelos personalizados no Neptune ML

É necessário um script de transformação para aproveitar o fluxo de trabalho incremental do Neptune ML para inferência de modelos em grafos em evolução sem treinar novamente o modelo. Mesmo que todos os artefatos necessários para a implantação do modelo sejam gerados pelo script de treinamento, você ainda precisará fornecer um script de transformação se quiser gerar modelos atualizados sem treinar novamente o modelo.

> **Note**
>
> No momento, a inferência indutiva em tempo real não é compatível com modelos personalizados.

O script de transformação de modelos personalizados deve ser um script Python executável, como o exemplo de script transform.py do kit de ferramentas do Neptune ML. Como esse script é invocado durante o treinamento de modelos sem argumentos de linha de comando, todos os argumentos de linha de comando aceitos pelo script devem ter padrões.

O script é executado em uma instância de treinamento do SageMaker com uma sintaxe como esta:

```
python3 (your transform script entry point)
```

O script de transformação precisará de várias informações, como:

- A localização dos dados de grafos processados.
- O local onde os parâmetros do modelo são salvos e onde os novos artefatos do modelo devem ser salvos.

- Os dispositivos disponíveis na instância.
- Os hiperparâmetros que geraram o melhor modelo.

Essas entradas são obtidas usando as funções de utilitário do Neptune ML que o script pode chamar. Veja o exemplo do script transform.py do kit de ferramentas para ver exemplos de como fazer isso.

O script deve salvar as incorporações do nó, os mapeamentos de ID de nó e quaisquer outros artefatos necessários para a implantação do modelo em cada tarefa. Consulte a documentação dos artefatos do modelo para obter mais informações sobre os artefatos do modelo necessários para diferentes tarefas do Neptune ML.

Arquivo **`model-hpo-configuration.json`** personalizado no Neptune ML

O arquivo `model-hpo-configuration.json` define hiperparâmetros para o modelo personalizado. Ele está no mesmo formato que o arquivo `model-hpo-configuration.json` usado com os modelos integrados do Neptune ML e tem precedência sobre a versão gerada automaticamente pelo Neptune ML e carregada no local dos dados processados.

Ao adicionar um novo hiperparâmetro ao modelo, você também deve adicionar uma entrada para o hiperparâmetro nesse arquivo para que o hiperparâmetro seja transmitido ao script de treinamento.

Será necessário fornecer um intervalo para um hiperparâmetro se quiser que ele seja ajustável e defini-lo como parâmetro `tier-1`, `tier-2` ou `tier-3`. O hiperparâmetro será ajustado se o número total de trabalhos de treinamento configurados permitir o ajuste de hiperparâmetros em seu nível. Para um parâmetro não ajustável, é necessário fornecer um valor padrão e adicionar o hiperparâmetro à seção `fixed-param` do arquivo. Veja o exemplo de arquivo `model-hpo-configuration.json` do kit de ferramentas para ver um exemplo de como fazer isso.

Você também deve fornecer a definição de métrica que a tarefa de otimização de hiperparâmetros do SageMaker usará para avaliar os modelos candidatos treinados. Para fazer isso, adicione um objeto JSON `eval_metric` ao arquivo `model-hpo-configuration.json` da seguinte forma:

```
"eval_metric": {
  "tuning_objective": {
      "MetricName": "(metric_name)",
      "Type": "Maximize"
  },
  "metric_definitions": [
    {
```

```
      "Name": "(metric_name)",
      "Regex": "(metric regular expression)"
    }
  ]
},
```

A matriz `metric_definitions` no objeto `eval_metric` lista objetos de definição de cada métrica que você deseja que o SageMaker extraia da instância de treinamento. Cada objeto de definição de métrica tem uma chave `Name` que permite fornecer um nome para a métrica (como “precisão”, “f1”, etc.). A chave `Regex` permite fornecer uma string de expressão regular que corresponda à forma como essa métrica específica é impressa nos logs de treinamento. Consulte a página SageMaker HyperParameter Tuning para obter mais detalhes sobre como definir métricas.

O objeto `tuning_objective` em `eval_metric` então permite especificar quais métricas em `metric_definitions` devem ser usadas como a métrica de avaliação que serve como métrica objetiva para otimização de hiperparâmetros. O valor para o `MetricName` deve corresponder ao valor de um `Name` em uma das definições em `metric_definitions`. O valor de `Type` deve ser “Maximizar” ou “Minimizar”, dependendo se a métrica deve ser interpretada como maior é melhor (como “precisão”) ou menos é melhor (como “erro quadrático médio”).

Erros nessa seção do arquivo `model-hpo-configuration.json` podem gerar falhas no trabalho da API de treinamento de modelos do Neptune ML, porque o trabalho de ajuste de hiperparâmetros do SageMaker não poderá selecionar o melhor modelo.

Teste local da implementação do modelo personalizado no Neptune ML

É possível usar o ambiente Conda do kit de ferramentas do Neptune ML para executar o código localmente a fim de testar e validar o modelo. Se você estiver desenvolvendo em uma instância do Neptune Notebook, esse ambiente Conda será pré-instalado na instância de bloco de anotações do Neptune. Se você estiver desenvolvendo em uma instância diferente, precisará seguir as instruções de configuração local no kit de ferramentas do Neptune ML.

O ambiente Conda reproduz com precisão o ambiente em que o modelo será executado quando você chamar a API de treinamento de modelos. Todos os exemplos de script de treinamento e scripts de transformação permitem transmitir um sinalizador `--local` de linha de comando para executar os scripts em um ambiente local a fim de facilitar a depuração. Essa é uma prática recomendada ao desenvolver o próprio modelo, pois permite testar de forma interativa e iterativa a implementação do modelo. Durante o treinamento de modelos no ambiente de treinamento de produção do Neptune ML, esse parâmetro é omitido.

Criar um endpoint de inferência para consulta

Um endpoint de inferência permite consultar um modelo específico que o processo de treinamento de modelos criou. O endpoint se conecta ao modelo de melhor desempenho de um tipo específico que o processo de treinamento conseguiu gerar. O endpoint então pode aceitar consultas do Gremlin do Neptune e gerar as previsões desse modelo para entradas nas consultas. Depois de criar um endpoint de referência, ele permanece ativo até a exclusão.

Gerenciar endpoints de inferência para o Neptune ML

Depois de concluir o treinamento de modelos em dados que você exportou do Neptune, você poderá criar um endpoint de inferência usando um comando `curl` (ou `awscurl`) como o seguinte:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/endpoints
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "mlModelTrainingJobId": "(the model-training job-id of a completed job)"
      }'
```

Também é possível criar um endpoint de inferência a partir de um modelo criado por um trabalho de transformação de modelos concluído, da mesma forma:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/endpoints
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "mlModelTransformJobId": "(the model-transform job-id of a completed job)"
      }'
```

Os detalhes de como usar esses comandos são explicados em O comando endpoints, junto com informações sobre como obter o status de um endpoint, excluir um endpoint e listar todos os endpoints de inferência.

Consultas de inferência no Neptune ML

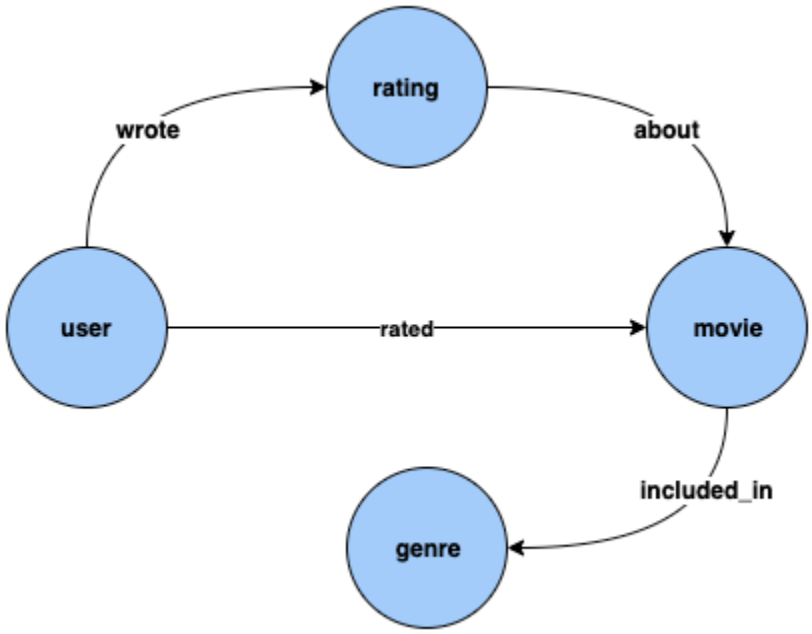
É possível usar o Gremlin ou o SPARQL para consultar um endpoint de inferência do Neptune ML. No entanto, a inferência indutiva em tempo real no momento só é aceita para consultas do Gremlin.

Consultas de inferência do Gremlin no Neptune ML

Conforme descrito em Capacidades do Neptune ML, o Neptune ML é compatível com modelos de treinamento que podem realizar os seguintes tipos de tarefa de inferência:

- Classificação de nós: prevê o atributo categórico de uma propriedade de vértice.
- Regressão de nós: prevê uma propriedade numérica de um vértice.
- Classificação de bordas: prevê o atributo categórico de uma propriedade de borda.
- Regressão de nós: prevê uma propriedade numérica de uma borda.
- Previsão de links: prevê nós de destino considerando-se um nó de origem e uma borda de saída, ou nós de origem considerando-se um nó de destino e uma borda de entrada.

Podemos ilustrar essas diferentes tarefas com exemplos que usam o conjunto de dados MovieLens 100k fornecido pela GroupLens Research. Esse conjunto de dados consiste em filmes, usuários e avaliações dos filmes feitas pelos usuários, a partir das quais criamos um grafo de propriedades como este:



Classificação de nós: no conjunto de dados acima, `Genre` é um tipo de vértice conectado ao tipo de vértice `Movie` por borda `included_in`. No entanto, se ajustarmos o conjunto de dados para tornar `Genre` um atributo categórico para o tipo de vértice `Movie`, o problema de inferir `Genre` para novos filmes adicionados ao nosso grafo de conhecimento poderá ser resolvido usando modelos de classificação de nós.

Regressão de nós: se considerarmos o tipo de vértice `Rating`, que tem propriedades, como `timestamp` e `score`, o problema de inferir o valor numérico `Score` para um `Rating` pode ser resolvido usando modelos de regressão de nós.

Classificação de bordas: da mesma forma, para uma borda `Rated`, se tivermos uma propriedade `Scale` que pode ter um dos valores, `Love`,`Like`, `Dislike`, `Neutral`, `Hate`, o problema de inferir `Scale` para a borda `Rated` para novos filmes/classificações pode ser resolvido usando modelos de classificação de bordas.

Regressão de bordas: da mesma forma, para a mesma borda `Rated`, se tivermos uma propriedade `Score` que contém um valor numérico para a classificação, isso pode ser inferido a partir de modelos de regressão de bordas.

Previsão de links: problemas, por exemplo, encontrar os dez principais usuários com maior probabilidade de avaliar um determinado filme ou encontrar os dez melhores filmes que um determinado usuário tem maior probabilidade de avaliar se enquadram na previsão de links.

> **Note**
>
> Para casos de uso do Neptune ML, temos um conjunto muito rico de cadernos projetados para oferecer uma compreensão prática de cada caso de uso. Você pode criar esses cadernos junto com seu cluster do Neptune ao usar o modelo do Neptune ML AWS CloudFormation para criar um cluster do Neptune ML. Esses cadernos também estão disponíveis no github.

Tópicos

Predicados do Neptune ML usados em consultas de inferência do Gremlin

`Neptune#ml.deterministic`

Esse predicado é uma opção para consultas de inferência indutiva, ou seja, para consultas que incluem o predicado `Neptune#ml.inductiveInference`.

Ao usar a inferência indutiva, o mecanismo do Neptune cria o subgrafo apropriado para avaliar o modelo de GNN treinado, e os requisitos desse subgrafo dependem dos parâmetros do modelo final. Especificamente, o parâmetro `num-layer` determina o número de saltos transversais dos nós ou das bordas de destino, e o parâmetro `fanouts` especifica quantos vizinhos devem ser incluídos na amostra em cada salto (consulte os parâmetros do HPO).

Por padrão, as consultas de inferência indutiva são executadas no modo não determinístico, no qual o Neptune constrói a vizinhança aleatoriamente. Ao fazer previsões, essa amostragem normal de vizinhos aleatórios às vezes gera previsões diferentes.

Quando você inclui `Neptune#ml.deterministic` em uma consulta de inferência indutiva, o mecanismo do Neptune tenta realizar uma amostra de vizinhos de forma determinística para que várias invocações da mesma consulta retornem sempre os mesmos resultados. No entanto, não se pode garantir que os resultados sejam completamente determinísticos, pois as alterações no grafo e nos artefatos subjacentes dos sistemas distribuídos ainda podem introduzir flutuações.

Você inclui o predicado `Neptune#ml.deterministic` em uma consulta como esta:

```
.with("Neptune#ml.deterministic")
```

Se o predicado `Neptune#ml.deterministic` for incluído em uma consulta que também não inclua `Neptune#ml.inductiveInference`, ele será simplesmente ignorado.

`Neptune#ml.disableInductiveInferenceMetadataCache`

Esse predicado é uma opção para consultas de inferência indutiva, ou seja, para consultas que incluem o predicado `Neptune#ml.inductiveInference`.

Para consultas de inferência indutiva, o Neptune usa um arquivo de metadados armazenado no Amazon S3 para decidir o número de saltos e o fanout ao construir a vizinhança. O Neptune normalmente armazena em cache esses metadados do modelo para evitar a busca repetida do arquivo no Amazon S3. O armazenamento em cache pode ser desativado incluindo o predicado `Neptune#ml.disableInductiveInferenceMetadataCache` na consulta. Embora possa ser mais lento para o Neptune buscar os metadados diretamente no Amazon S3, é útil quando o

endpoint do SageMaker foi atualizado após um novo treinamento ou transformação e o cache ficar obsoleto.

Você inclui o predicado `Neptune#ml.disableInductiveInferenceMetadataCache` em uma consulta como esta:

```
.with("Neptune#ml.disableInductiveInferenceMetadataCache")
```

Veja um exemplo de consulta em um caderno Jupyter:

```
%%gremlin
g.with("Neptune#ml.endpoint", "ep1")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .with("Neptune#ml.disableInductiveInferenceMetadataCache")
 .V('101').properties("rating")
 .with("Neptune#ml.regression")
 .with("Neptune#ml.inductiveInference")
```

`Neptune#ml.endpoint`

O predicado `Neptune#ml.endpoint` é usado em uma etapa `with()` para especificar o endpoint de inferência, se necessário:

```
 .with("Neptune#ml.endpoint", "the model's SageMaker inference endpoint")
```

Você pode identificar o endpoint pelo `id` ou pelo URL. Por exemplo:

```
 .with( "Neptune#ml.endpoint", "node-classification-movie-lens-endpoint" )
```

Ou:

```
 .with( "Neptune#ml.endpoint", "https://runtime.sagemaker.us-east-1.amazonaws.com/
endpoints/node-classification-movie-lens-endpoint/invocations" )
```

> **Note**
>
> Se você definir o parâmetro `neptune_ml_endpoint` no grupo de parâmetros do cluster de banco de dados do Neptune como o URL ou o `id` do endpoint, não precisará incluir o predicado `Neptune#ml.endpoint` em cada consulta.

`Neptune#ml.iamRoleArn`

`Neptune#ml.iamRoleArn` é usado em uma etapa `with()` para especificar o ARN do perfil do IAM de execução do SageMaker, se necessário:

```
.with("Neptune#ml.iamRoleArn", "the ARN for the SageMaker execution IAM role")
```

Para obter informações sobre como criar o perfil do IAM de execução do SageMaker, consulte Criar um perfil `NeptuneSageMakerIAMRole` personalizado.

> **Note**
>
> Se você definir o parâmetro `neptune_ml_iam_role` no grupo de parâmetros do cluster de banco de dados do Neptune como o ARN do perfil do IAM de execução do SageMaker, não precisará incluir o predicado `Neptune#ml.iamRoleArn` em cada consulta.

Neptune#ml.inductiveInference

A inferência transdutiva é habilitada por padrão no Gremlin. Para fazer uma consulta de inferência indutiva em tempo real, inclua o predicado `Neptune#ml.inductiveInference` desta forma:

```
.with("Neptune#ml.inductiveInference")
```

Se seu grafo for dinâmico, a inferência indutiva geralmente será a melhor escolha, mas se seu grafo for estático, a inferência transdutiva será mais rápida e eficiente.

`Neptune#ml.limit`

Opcionalmente, o predicado `Neptune#ml.limit` limita o número de resultados gerados por entidade:

```
.with( "Neptune#ml.limit", 2 )
```

Por padrão, o limite é 1 e o número máximo que pode ser definido é 100.

`Neptune#ml.threshold`

Opcionalmente, o predicado `Neptune#ml.threshold` estabelece um limite de corte para as pontuações dos resultados:

```
.with( "Neptune#ml.threshold", 0.5D )
```

Isso permite que você descarte todos os resultados com pontuações abaixo do limite especificado.

Neptune#ml.classification

O predicado `Neptune#ml.classification` é anexado à etapa `properties()` para estabelecer que as propriedades precisam ser obtidas do endpoint do SageMaker do modelo de classificação de nós:

```
.properties( "property key of the node classification
model" ).with( "Neptune#ml.classification" )
```

Neptune#ml.regression

O predicado `Neptune#ml.regression` é anexado à etapa `properties()` para estabelecer que as propriedades precisam ser obtidas do endpoint do SageMaker do modelo de regressão de nós:

```
.properties( "property key of the node regression
model" ).with( "Neptune#ml.regression" )
```

Neptune#ml.prediction

O predicado `Neptune#ml.prediction` é anexado às etapas `in()` e `out()` para estabelecer que se trata de uma consulta de previsão de links:

```
.in("edge label of the link prediction
model").with("Neptune#ml.prediction").hasLabel("target node label")
```

Neptune#ml.score

O predicado `Neptune#ml.score` é usado em consultas de classificação de nós ou bordas do Gremlin para obter uma pontuação de confiança de machine learning. O predicado `Neptune#ml.score` deve ser transmitido junto com o predicado da consulta na etapa `properties()` para obter uma pontuação de confiança de ML para consultas de classificação de nós ou bordas.

É possível encontrar um exemplo de classificação de nós com outros exemplos de classificação de nós e um exemplo de classificação de bordas na seção de classificação de bordas.

Consultas de classificação de nós do Gremlin no Neptune ML

Para consultas de classificação de nós do Gremlin no Neptune ML:

- O modelo é treinado em uma propriedade dos vértices. O conjunto de valores exclusivos dessa propriedade é chamado de conjunto de classes de nós ou, simplesmente, classes.
- A classe do nó ou o valor da propriedade categórica da propriedade de um vértice pode ser inferido do modelo de classificação de nós. Isso é útil quando essa propriedade ainda não está anexada ao vértice.
- Para buscar uma ou mais classes de um modelo de classificação de nós, você precisa usar a etapa `with()` com o predicado `Neptune#ml.classification` para configurar a etapa `properties()`. O formato de saída é semelhante ao esperado se essas fossem propriedades de vértice.

> **Note**
>
> A classificação de nós só funciona com valores de propriedades de string. Isso significa que valores de propriedades numéricas, como `0` ou 1, não são compatíveis, embora a string seja equivalente a `"0"` e `"1"`. Da mesma forma, os valores de propriedade booliana `true` e `false` não funcionam, mas `"true"` e `"false"` funcionam.

Veja a seguir um exemplo de consulta de classificação de nós:

```
g.with( "Neptune#ml.endpoint","node-classification-movie-lens-endpoint" )
 .with( "Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role" )
 .with( "Neptune#ml.limit", 2 )
 .with( "Neptune#ml.threshold", 0.5D )
 .V( "movie_1", "movie_2", "movie_3" )
 .properties("genre").with("Neptune#ml.classification")
```

O resultado dessa consulta seria algo semelhante ao seguinte:

```
==>vp[genre->Action]
==>vp[genre->Crime]
==>vp[genre->Comedy]
```

Na consulta acima, as etapas `V()` e `properties()` são usadas da seguinte forma:

A etapa `V()` contém o conjunto de vértices para os quais você deseja buscar as classes do modelo de classificação de nós:

```
  .V( "movie_1", "movie_2", "movie_3" )
```

A etapa `properties()` contém a chave na qual o modelo foi treinado e tem `.with("Neptune#ml.classification")` para indicar que se trata de uma consulta de inferência de ML de classificação de nós.

Atualmente, não são aceitas várias chaves de propriedade em uma etapa `properties().with("Neptune#ml.classification")`. Por exemplo, a seguinte consulta gera uma exceção:

```
g.with("Neptune#ml.endpoint", "node-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V( "movie_1", "movie_2", "movie_3" )
 .properties("genre", "other_label").with("Neptune#ml.classification")
```

Para ver a mensagem de erro específica, consulte a lista de exceções do Neptune ML.

Uma etapa `properties().with("Neptune#ml.classification")` pode ser usada em combinação com qualquer uma das seguintes etapas:

- `value()`
- `value().is()`
- `hasValue()`
- `has(value,"")`
- `key()`
- `key().is()`
- `hasKey()`
- `has(key,"")`
- `path()`

Outras consultas de classificação de nós

Se o endpoint de inferência e o perfil do IAM correspondente tiverem sido salvos no seu grupo de parâmetros do cluster de banco de dados, uma consulta de classificação de nós poderá ser tão simples quanto esta:

```
g.V("movie_1", "movie_2",
 "movie_3").properties("genre").with("Neptune#ml.classification")
```

Você pode misturar propriedades e classes de vértices em uma consulta usando a etapa `union()`:

```
g.with("Neptune#ml.endpoint","node-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V( "movie_1", "movie_2", "movie_3" )
 .union(
   properties("genre").with("Neptune#ml.classification"),
   properties("genre")
 )
```

Também é possível fazer uma consulta ilimitada, como esta:

```
g.with("Neptune#ml.endpoint","node-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V()
 .properties("genre").with("Neptune#ml.classification")
```

Você pode recuperar as classes de nós junto com os vértices usando a etapa `select()` junto com a etapa `as()`:

```
g.with("Neptune#ml.endpoint","node-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V( "movie_1", "movie_2", "movie_3" ).as("vertex")
 .properties("genre").with("Neptune#ml.classification").as("properties")
 .select("vertex","properties")
```

Também é possível filtrar por classes de nós, conforme ilustrado nestes exemplos:

```
g.with("Neptune#ml.endpoint", "node-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V( "movie_1", "movie_2", "movie_3" )
```

```
  .properties("genre").with("Neptune#ml.classification")
  .has(value, "Horror")

g.with("Neptune#ml.endpoint","node-classification-movie-lens-endpoint")
  .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
  .V( "movie_1", "movie_2", "movie_3" )
  .properties("genre").with("Neptune#ml.classification")
  .has(value, P.eq("Action"))

g.with("Neptune#ml.endpoint","node-classification-movie-lens-endpoint")
  .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
  .V( "movie_1", "movie_2", "movie_3" )
  .properties("genre").with("Neptune#ml.classification")
  .has(value, P.within("Action", "Horror"))
```

Você pode obter uma pontuação de confiança na classificação de nós usando o predicado `Neptune#ml.score`:

```
 g.with("Neptune#ml.endpoint","node-classification-movie-lens-endpoint")
  .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
  .V( "movie_1", "movie_2", "movie_3" )
  .properties("genre", "Neptune#ml.score").with("Neptune#ml.classification")
```

A resposta seria semelhante a esta:

```
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.01234567]
==>vp[genre->Crime]
==>vp[Neptune#ml.score->0.543210]
==>vp[genre->Comedy]
==>vp[Neptune#ml.score->0.10101]
```

Usar inferência indutiva em uma consulta de classificação de nós

Suponha que você adicione um novo nó a um grafo existente, em um caderno Jupyter, da seguinte forma:

```
%%gremlin
g.addV('label1').property(id,'101').as('newV')
  .V('1').as('oldV1')
  .V('2').as('oldV2')
  .addE('eLabel1').from('newV').to('oldV1')
```

```
  .addE('eLabel2').from('oldV2').to('newV')
```

Depois, você poderia usar uma consulta de inferência indutiva para obter um gênero e uma pontuação de confiança que refletissem o novo nó:

```
%%gremlin
g.with("Neptune#ml.endpoint", "nc-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .V('101').properties("genre", "Neptune#ml.score")
 .with("Neptune#ml.classification")
 .with("Neptune#ml.inductiveInference")
```

No entanto, se você executou a consulta várias vezes, poderá obter resultados um pouco diferentes:

```
# First time
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.12345678]

# Second time
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.21365921]
```

Você pode tornar a mesma consulta determinística:

```
%%gremlin
g.with("Neptune#ml.endpoint", "nc-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .V('101').properties("genre", "Neptune#ml.score")
 .with("Neptune#ml.classification")
 .with("Neptune#ml.inductiveInference")
 .with("Neptune#ml.deterministic")
```

Nesse caso, os resultados seriam praticamente os mesmos todas as vezes:

```
# First time
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.12345678]
# Second time
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.12345678]
```

Consultas de regressão de nós do Gremlin no Neptune ML

A regressão de nós é semelhante à classificação de nós, exceto que o valor inferido do modelo de regressão para cada nó é numérico. É possível usar as mesmas consultas do Gremlin para regressão e classificação de nós, exceto pelas seguintes diferenças:

- Novamente, no Neptune ML, os nós se referem a vértices.
- A etapa `properties()` assume a forma `properties().with("Neptune#ml.regression")` em vez de `properties().with("Neptune#ml.classification")`.
- Os predicados `"Neptune#ml.limit` e `"Neptune#ml.threshold"` não são aplicáveis.
- Ao filtrar o valor, você precisa especificar um valor numérico.

Veja a seguir um exemplo de consulta de classificação de vértices:

```
g.with("Neptune#ml.endpoint","node-regression-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::0123456789:role/sagemaker-role")
 .V("movie_1","movie_2","movie_3")
 .properties("revenue").with("Neptune#ml.regression")
```

Você pode filtrar o valor inferido usando um modelo de regressão, conforme ilustrado nos seguintes exemplos:

```
g.with("Neptune#ml.endpoint","node-regression-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("movie_1","movie_2","movie_3")
 .properties("revenue").with("Neptune#ml.regression")
 .value().is(P.gte(1600000))

g.with("Neptune#ml.endpoint","node-regression-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("movie_1","movie_2","movie_3")
 .properties("revenue").with("Neptune#ml.regression")
 .hasValue(P.lte(1600000D))
```

Usar inferência indutiva em uma consulta de regressão de nós

Suponha que você adicione um novo nó a um grafo existente, em um caderno Jupyter, da seguinte forma:

```
%%gremlin
g.addV('label1').property(id,'101').as('newV')
 .V('1').as('oldV1')
 .V('2').as('oldV2')
 .addE('eLabel1').from('newV').to('oldV1')
 .addE('eLabel2').from('oldV2').to('newV')
```

Depois, você pode usar uma consulta de inferência indutiva para obter uma classificação que leve em conta o novo nó:

```
%%gremlin
g.with("Neptune#ml.endpoint", "nr-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .V('101').properties("rating")
 .with("Neptune#ml.regression")
 .with("Neptune#ml.inductiveInference")
```

Como a consulta não é determinística, ela poderá gerar resultados um pouco diferentes se você a executar várias vezes, com base na vizinhança:

```
# First time
==>vp[rating->9.1]

# Second time
==>vp[rating->8.9]
```

Se precisar de resultados mais consistentes, você poderá tornar a consulta determinística:

```
%%gremlin
g.with("Neptune#ml.endpoint", "nc-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .V('101').properties("rating")
 .with("Neptune#ml.regression")
 .with("Neptune#ml.inductiveInference")
 .with("Neptune#ml.deterministic")
```

Agora, os resultados serão praticamente os mesmos todas as vezes:

```
# First time
==>vp[rating->9.1]
```

```
# Second time
==>vp[rating->9.1]
```

Consultas de classificação de bordas do Gremlin no Neptune ML

Para consultas de classificação de bordas do Gremlin no Neptune ML:

- O modelo é treinado em uma propriedade das bordas. O conjunto de valores exclusivos dessa propriedade é chamado de conjunto de classes.
- O valor da classe ou da propriedade categórica de uma borda pode ser inferido do modelo de classificação de bordas, o que é útil quando essa propriedade ainda não está anexada à borda.
- Para buscar uma ou mais classes de um modelo de classificação de bordas, é necessário usar a etapa `with()` com o predicado, `"Neptune#ml.classification"` para configurar a etapa `properties()`. O formato de saída é semelhante ao esperado se essas fossem propriedades de borda.

> **Note**
>
> A classificação de bordas só funciona com valores de propriedades de string. Isso significa que valores de propriedades numéricas, como `0` ou `1`, não são compatíveis, embora a string seja equivalente a `"0"` e `"1"`. Da mesma forma, os valores de propriedade booliana `true` e `false` não funcionam, mas `"true"` e `"false"` funcionam.

Aqui está um exemplo de consulta de classificação de bordas que solicita uma pontuação de confiança usando o predicado `Neptune#ml.score`:

```
g.with("Neptune#ml.endpoint","edge-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .E("relationship_1","relationship_2","relationship_3")
 .properties("knows_by", "Neptune#ml.score").with("Neptune#ml.classification")
```

A resposta seria semelhante a esta:

```
==>p[knows_by->"Family"]
==>p[Neptune#ml.score->0.01234567]
==>p[knows_by->"Friends"]
```

```
==>p[Neptune#ml.score->0.543210]
==>p[knows_by->"Colleagues"]
==>p[Neptune#ml.score->0.10101]
```

Sintaxe de uma consulta de classificação de bordas do Gremlin

Para um grafo simples em que `User` é o nó principal e final e `Relationship` é a borda que os conecta, um exemplo de consulta de classificação de bordas é:

```
g.with("Neptune#ml.endpoint","edge-classification-social-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .E("relationship_1","relationship_2","relationship_3")
 .properties("knows_by").with("Neptune#ml.classification")
```

O resultado dessa consulta seria algo semelhante ao seguinte:

```
==>p[knows_by->"Family"]
==>p[knows_by->"Friends"]
==>p[knows_by->"Colleagues"]
```

Na consulta acima, as etapas `E()` e `properties()` são usadas da seguinte forma:

- A etapa `E()` contém o conjunto de bordas para os quais você deseja buscar as classes do modelo de classificação de bordas:

  ```
  .E("relationship_1","relationship_2","relationship_3")
  ```

- A etapa `properties()` contém a chave na qual o modelo foi treinado e tem `.with("Neptune#ml.classification")` para indicar que se trata de uma consulta de inferência de ML de classificação de bordas.

Atualmente, não são aceitas várias chaves de propriedade em uma etapa `properties().with("Neptune#ml.classification")`. Por exemplo, a consulta a seguir ocasiona o lançamento de uma exceção:

```
g.with("Neptune#ml.endpoint","edge-classification-social-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .E("relationship_1","relationship_2","relationship_3")
 .properties("knows_by", "other_label").with("Neptune#ml.classification")
```

Sobre mensagens de erro específicas, consulte Lista de exceções para consultas de inferência do Gremlin no Neptune ML.

Uma etapa `properties().with("Neptune#ml.classification")` pode ser usada em combinação com qualquer uma das seguintes etapas:

- `value()`
- `value().is()`
- `hasValue()`
- `has(value,"")`
- `key()`
- `key().is()`
- `hasKey()`
- `has(key,"")`
- `path()`

Usar inferência indutiva em uma consulta de classificação de bordas

Suponha que você adicione uma nova borda a um grafo existente, em um caderno Jupyter, da seguinte forma:

```
%%gremlin
g.V('1').as('fromV')
.V('2').as('toV')
.addE('eLabel1').from('fromV').to('toV').property(id, 'e101')
```

Depois, você poderia usar uma consulta de inferência indutiva para obter uma escala que levasse em conta a nova borda:

```
%%gremlin
g.with("Neptune#ml.endpoint", "ec-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .E('e101').properties("scale", "Neptune#ml.score")
 .with("Neptune#ml.classification")
 .with("Neptune#ml.inductiveInference")
```

Como a consulta não é determinística, os resultados seriam um pouco diferentes se você a executasse várias vezes, com base na vizinhança aleatória:

```
# First time
==>vp[scale->Like]
==>vp[Neptune#ml.score->0.12345678]

# Second time
==>vp[scale->Like]
==>vp[Neptune#ml.score->0.21365921]
```

Se precisar de resultados mais consistentes, você poderá tornar a consulta determinística:

```
%%gremlin
g.with("Neptune#ml.endpoint", "ec-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .E('e101').properties("scale", "Neptune#ml.score")
 .with("Neptune#ml.classification")
 .with("Neptune#ml.inductiveInference")
 .with("Neptune#ml.deterministic")
```

Agora, os resultados serão mais ou menos os mesmos toda vez que você executar a consulta:

```
# First time
==>vp[scale->Like]
==>vp[Neptune#ml.score->0.12345678]

# Second time
==>vp[scale->Like]
==>vp[Neptune#ml.score->0.12345678]
```

Consultas de regressão de bordas do Gremlin no Neptune ML

A regressão de bordas é semelhante à classificação de bordas, exceto que o valor inferido do modelo de ML é numérico. Para regressão de bordas, o Neptune ML é compatível com as mesmas consultas de classificação.

Os principais pontos a serem observados são:

- Você precisa usar o predicado de ML `"Neptune#ml.regression"` para configurar a etapa `properties()` para esse caso de uso.
- Os predicados `"Neptune#ml.limit"` e `"Neptune#ml.threshold"` não são aplicáveis nesse caso de uso.

- Para filtrar o valor, você precisa especificar o valor como numérico.

Sintaxe de uma consulta de regressão de bordas do Gremlin

Para um grafo simples em que `User` está o nó principal, `Movie` o nó final e `Rated` é a borda que os conecta, aqui está um exemplo de consulta de regressão de bordas que encontra o valor da classificação numérica, chamado de pontuação aqui, da borda `Rated`:

```
g.with("Neptune#ml.endpoint","edge-regression-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .E("rating_1","rating_2","rating_3")
 .properties("score").with("Neptune#ml.regression")
```

Também é possível filtrar por um valor inferido do modelo de regressão de ML. Para as bordas `Rated` existentes (de `User` a `Movie`) identificadas por `"rating_1"`, `"rating_2"` e `"rating_3"`, em que a propriedade de borda `Score` não está presente para essas classificações, é possível usar uma consulta como a seguinte para inferir `Score` para as bordas em que ela é maior ou igual a 9:

```
g.with("Neptune#ml.endpoint","edge-regression-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .E("rating_1","rating_2","rating_3")
 .properties("score").with("Neptune#ml.regression")
 .value().is(P.gte(9))
```

Usar inferência indutiva em uma consulta de regressão de bordas

Suponha que você adicione uma nova borda a um grafo existente, em um caderno Jupyter, da seguinte forma:

```
%%gremlin
g.V('1').as('fromV')
.V('2').as('toV')
.addE('eLabel1').from('fromV').to('toV').property(id, 'e101')
```

Depois, você pode usar uma consulta de inferência indutiva para obter uma pontuação que leve em conta a nova borda:

```
%%gremlin
g.with("Neptune#ml.endpoint", "er-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
```

```
 .E('e101').properties("score")
 .with("Neptune#ml.regression")
 .with("Neptune#ml.inductiveInference")
```

Como a consulta não é determinística, os resultados seriam um pouco diferentes se você a executasse várias vezes, com base na vizinhança aleatória:

```
# First time
==>ep[score->96]

# Second time
==>ep[score->91]
```

Se precisar de resultados mais consistentes, você poderá tornar a consulta determinística:

```
%%gremlin
g.with("Neptune#ml.endpoint", "er-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .E('e101').properties("score")
 .with("Neptune#ml.regression")
 .with("Neptune#ml.inductiveInference")
 .with("Neptune#ml.deterministic")
```

Agora, os resultados serão mais ou menos os mesmos toda vez que você executar a consulta:

```
# First time
==>ep[score->96]

# Second time
==>ep[score->96]
```

Consultas de previsão de links do Gremlin usando modelos de previsão de links no Neptune ML

Os modelos de previsão de links podem resolver problemas, como os seguintes:

- Previsão do nó principal: considerando um tipo de vértice e borda, a quais vértices esse vértice provavelmente se vinculará?
- Previsão do nó final: considerando um rótulo de vértice e borda, a quais vértices esse vértice provavelmente se vinculará?

> **Note**
>
> A previsão de borda ainda não é aceita no Neptune ML.

Para os exemplos abaixo, considere um grafo simples com os vértices `User` e `Movie` que estão vinculados pela borda `Rated`.

Aqui está um exemplo de consulta de previsão do nó principal, usada para prever os cinco principais usuários com maior probabilidade de avaliar os filmes, `"movie_1"`, `"movie_2"` e `"movie_3"`.

```
g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .with("Neptune#ml.limit", 5)
 .V("movie_1", "movie_2", "movie_3")
 .in("rated").with("Neptune#ml.prediction").hasLabel("user")
```

Aqui está uma similar para previsão de nó final, usada para prever os cinco principais filmes que o usuário `"user_1"` provavelmente avaliará:

```
g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("user_1")
 .out("rated").with("Neptune#ml.prediction").hasLabel("movie")
```

Tanto o rótulo da borda quanto o rótulo do vértice previsto são obrigatórios. Se um deles for omitido, uma exceção será lançada. Por exemplo, a seguinte consulta sem um rótulo de vértice previsto gera uma exceção:

```
g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("user_1")
 .out("rated").with("Neptune#ml.prediction")
```

Da mesma forma, a seguinte consulta sem um rótulo de borda gera uma exceção:

```
g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("user_1")
 .out().with("Neptune#ml.prediction").hasLabel("movie")
```

Para ver a mensagem de erro específica, consulte a lista de exceções do Neptune ML.

Outras consultas de previsão de links

Você pode usar a etapa `select()` com a etapa `as()` para gerar os vértices previstos junto com os vértices de entrada:

```
g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("movie_1").as("source")
 .in("rated").with("Neptune#ml.prediction").hasLabel("user").as("target")
 .select("source","target")

g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("user_1").as("source")
 .out("rated").with("Neptune#ml.prediction").hasLabel("movie").as("target")
 .select("source","target")
```

É possível fazer consultas ilimitadas, como estas:

```
g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("user_1")
 .out("rated").with("Neptune#ml.prediction").hasLabel("movie")

g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("movie_1")
 .in("rated").with("Neptune#ml.prediction").hasLabel("user")
```

Usar inferência indutiva em uma consulta de previsão de links

Suponha que você adicione um novo nó a um grafo existente, em um caderno Jupyter, da seguinte forma:

```
%%gremlin
g.addV('label1').property(id,'101').as('newV1')
 .addV('label2').property(id,'102').as('newV2')
 .V('1').as('oldV1')
 .V('2').as('oldV2')
 .addE('eLabel1').from('newV1').to('oldV1')
```

```
  .addE('eLabel2').from('oldV2').to('newV2')
```

Você pode então usar uma consulta de inferência indutiva para prever o nó principal, levando em consideração o novo nó:

```
%%gremlin
g.with("Neptune#ml.endpoint", "lp-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .V('101').out("eLabel1")
 .with("Neptune#ml.prediction")
 .with("Neptune#ml.inductiveInference")
 .hasLabel("label2")
```

Resultado:

```
==>V[2]
```

Da mesma forma, você pode usar uma consulta de inferência indutiva para prever o nó final, levando em consideração o novo nó:

```
%%gremlin
g.with("Neptune#ml.endpoint", "lp-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .V('102').in("eLabel2")
 .with("Neptune#ml.prediction")
 .with("Neptune#ml.inductiveInference")
 .hasLabel("label1")
```

Resultado:

```
==>V[1]
```

Lista de exceções para consultas de inferência do Gremlin no Neptune ML

- **`BadRequestException`**: as credenciais do perfil fornecido não podem ser carregadas.

  Mensagem: `Unable to load credentials for role: the specified IAM Role ARN.`
- **`BadRequestException`**: o perfil do IAM especificado não está autorizado a invocar o endpoint do SageMaker.

Mensagem: `User: the specified IAM Role ARN is not authorized to perform: sagemaker:InvokeEndpoint on resource: the specified endpoint.`

- **`BadRequestException`**: o endpoint especificado não existe.

  Mensagem: `Endpoint the specified endpoint not found.`

- **`InternalFailureException`**: não é possível obter metadados de inferência indutiva em tempo real do Neptune ML do Amazon S3.

  Mensagem: `Unable to fetch Neptune ML - Real-Time Inductive Inference metadata from S3. Check the permissions of the S3 bucket or if the Neptune instance can connect to S3.`

- **`InternalFailureException`**: o Neptune ML não consegue encontrar o arquivo de metadados para inferência indutiva em tempo real no Amazon S3.

  Mensagem: `Neptune ML cannot find the metadata file for Real-Time Inductive Inference in S3.`

- **`InvalidParameterException`**: o endpoint especificado não é sintaticamente válido.

  Mensagem: `Invalid endpoint provided for external service query.`

- **`InvalidParameterException`**: o ARN do perfil do IAM de execução especificada do SageMaker não é sintaticamente válido.

  Mensagem: `Invalid IAM role ARN provided for external service query.`

- **`InvalidParameterException`**: várias chaves de propriedade são especificadas na etapa `properties()` de uma consulta.

  Mensagem: `ML inference queries are currently supported for one property key.`

- **`InvalidParameterException`**: vários rótulos de borda são especificados em uma consulta.

  Mensagem: `ML inference are currently supported only with one edge label.`

- **`InvalidParameterException`**: várias restrições de rótulos de vértice são especificadas em uma consulta.

  Mensagem: `ML inference are currently supported only with one vertex label constraint.`

- **InvalidParameterException**: os predicados `Neptune#ml.classification` e `Neptune#ml.regression` estão presentes na mesma consulta.

  Mensagem: `Both regression and classification ML predicates cannot be specified in the query.`
- **InvalidParameterException**: mais de um rótulo de borda foi especificado na etapa `in()` ou `out()` em uma consulta de previsão de links.

  Mensagem: `ML inference are currently supported only with one edge label.`
- **InvalidParameterException**: mais de uma chave de propriedade foi especificada com Neptune#ml.score.

  Mensagem: `Neptune ML inference queries are currently supported for one property key and one Neptune#ml.score property key.`
- **MissingParameterException**: o endpoint não foi especificado na consulta nem como um parâmetro de cluster de banco de dados.

  Mensagem: `No endpoint provided for external service query.`
- MissingParameterException: o perfil do IAM de execução do SageMaker não foi especificado na consulta nem como um parâmetro de cluster de banco de dados.

  Mensagem: `No IAM role ARN provided for external service query.`
- **MissingParameterException**: a chave da propriedade está ausente na etapa `properties()` de uma consulta.

  Mensagem: `Property key needs to be specified using properties() step for ML inference queries.`
- **MissingParameterException**: nenhum rótulo de borda foi especificado na etapa `in()` ou `out()` em uma consulta de previsão de links.

  Mensagem: `Edge label needs to be specified while using in() or out() step for ML inference queries.`
- **MissingParameterException**: nenhuma chave de propriedade foi especificada com Neptune#ml.score.

  Mensagem: `Property key needs to be specified along with Neptune#ml.score property key while using the properties() step for Neptune ML inference queries.`

- **UnsupportedOperationException**: a etapa `both()` é usada em uma consulta de previsão de links.

  Mensagem: `ML inference queries are currently not supported with both() step.`
- **UnsupportedOperationException**: nenhum rótulo de vértice previsto foi especificado na etapa `has()` com a etapa `in()` ou `out()` em uma consulta de previsão de links.

  Mensagem: `Predicted vertex label needs to be specified using has() step for ML inference queries.`
- **UnsupportedOperationException**: no momento, as consultas de inferência indutiva de ML do Gremlin não são compatíveis com etapas não otimizadas.

  Mensagem: `Neptune ML - Real-Time Inductive Inference queries are currently not supported with Gremlin steps which are not optimized for Neptune. Check the Neptune User Guide for a list of Neptune-optimized steps.`
- **UnsupportedOperationException**: no momento, as consultas de inferência do Neptune ML não são aceitas em uma etapa `repeat`.

  Mensagem: `Neptune ML inference queries are currently not supported inside a repeat step.`
- **UnsupportedOperationException**: no momento, não há compatibilidade com mais de uma consulta de inferência do Neptune ML por consulta do Gremlin.

  Mensagem: `Neptune ML inference queries are currently supported only with one ML inference query per gremlin query.`

Consultas de inferência do SPARQL no Neptune ML

O Neptune ML associa o grafo do RDF a um grafo de propriedades para modelar a tarefa de ML. No momento, ele é compatível com os seguintes casos de uso:

- Classificação de objetos: prevê o atributo categórico de um objeto.
- Regressão de objetos: prevê uma propriedade numérica de um objeto.
- Previsão de objetos: prevê um objeto considerando-se um assunto e um relacionamento.
- Previsão de assunto: prevê um assunto considerando-se um objeto e um relacionamento.

> **Note**
>
> O Neptune ML não é compatível com casos de uso de classificação e regressão de assuntos com SPARQL.

Predicados do Neptune ML usados em consultas de inferência do SPARQL

Os seguintes predicados são usados com inferência do SPARQL:

Predicado **`neptune-ml:timeout`**

Especifica o tempo limite para conexão com o servidor remoto. Não deve ser confundido com o tempo limite da solicitação de consulta, que é o tempo máximo que o servidor pode levar para atender a uma solicitação.

Observe que, se o tempo limite da consulta ocorrer antes do tempo limite do serviço especificado pelo predicado `neptune-ml:timeout`, a conexão do serviço também será cancelada.

Predicado **`neptune-ml:outputClass`**

O predicado `neptune-ml:outputClass` só é usado para definir a classe do objeto previsto para previsão de objetos ou do assunto previsto para a previsão do assunto.

Predicado **`neptune-ml:outputScore`**

O predicado `neptune-ml:outputScore` é um número positivo que representa a probabilidade de que a saída de um modelo de machine learning esteja correta.

Predicado **`neptune-ml:modelType`**

O predicado `neptune-ml:modelType` especifica o tipo de modelo de machine learning que está sendo treinado:

- `OBJECT_CLASSIFICATION`
- `OBJECT_REGRESSION`
- `OBJECT_PREDICTION`
- `SUBJECT_PREDICTION`

Predicado **`neptune-ml:input`**

O predicado `neptune-ml:input` se refere à lista de URIs usados como entradas para o Neptune ML.

Predicado **`neptune-ml:output`**

O predicado `neptune-ml:output` se refere à lista de conjuntos de vinculação em que o Neptune ML gera resultados.

Predicado **`neptune-ml:predicate`**

O predicado `neptune-ml:predicate` é usado de forma diferente dependendo da tarefa que estiver sendo realizada:

- Para previsão de objetos ou assuntos: define o tipo de predicado (o tipo de borda ou de relacionamento).
- Para classificação e regressão de objetos: define o literal (propriedade) que queremos prever.

Predicado **`neptune-ml:batchSize`**

O `neptune-ml:batchSize` especifica o tamanho da entrada para a chamada de serviço remoto.

Exemplos de classificação de objetos do SPARQL

Para classificação de objetos do SPARQL no Neptune ML, o modelo é treinado em um dos valores de predicados. Isso é útil quando esse predicado ainda não está presente em um assunto específico.

Somente valores de predicados categóricos podem ser inferidos usando o modelo de classificação de objetos.

A seguinte consulta busca prever o valor do predicado <http://www.example.org/team> para todas as entradas do tipo `foaf:Person`:

```
SELECT * WHERE { ?input a foaf:Person .
  SERVICE neptune-ml:inference {
    neptune-ml:config neptune-ml:modelType 'OBJECT_CLASSIFICATION' ;
                      neptune-ml:input ?input ;
                      neptune-ml:predicate <http://www.example.org/team> ;
                      neptune-ml:output ?output .
  }
}
```

Essa consulta pode ser personalizada da seguinte forma:

```
SELECT * WHERE { ?input a foaf:Person .
  SERVICE neptune-ml:inference {
    neptune-ml:config neptune-ml:endpoint 'node-prediction-account-balance-endpoint' ;
                      neptune-ml:iamRoleArn 'arn:aws:iam::0123456789:role/sagemaker-
role' ;

                      neptune-ml:batchSize "40"^^xsd:integer ;
                      neptune-ml:timeout "1000"^^xsd:integer ;

                      neptune-ml:modelType 'OBJECT_CLASSIFICATION' ;
                      neptune-ml:input ?input ;
                      neptune-ml:predicate <http://www.example.org/team> ;
                      neptune-ml:output ?output .
  }
}
```

Exemplos de regressão de objetos do SPARQL

A regressão de objetos é semelhante à classificação de objetos, exceto por um valor de predicado numérico inferido do modelo de regressão para cada nó. É possível usar as mesmas consultas do SPARQL para regressão de objetos e para classificação de objetos, com a exceção de que os predicados `the Neptune#ml.limit` e `Neptune#ml.threshold` não são aplicáveis.

A seguinte consulta busca prever o valor do predicado <http://www.example.org/accountbalance> para todas as entradas do tipo `foaf:Person`:

```
SELECT * WHERE { ?input a foaf:Person .
```

```
  SERVICE neptune-ml:inference {
    neptune-ml:config neptune-ml:modelType 'OBJECT_REGRESSION' ;
                      neptune-ml:input ?input ;
                      neptune-ml:predicate <http://www.example.org/accountbalance> ;
                      neptune-ml:output ?output .
  }
}
```

Essa consulta pode ser personalizada da seguinte forma:

```
SELECT * WHERE { ?input a foaf:Person .
  SERVICE neptune-ml:inference {
    neptune-ml:config neptune-ml:endpoint 'node-prediction-account-balance-endpoint' ;
                      neptune-ml:iamRoleArn 'arn:aws:iam::0123456789:role/sagemaker-
role' ;

                      neptune-ml:batchSize "40"^^xsd:integer ;
                      neptune-ml:timeout "1000"^^xsd:integer ;

                      neptune-ml:modelType 'OBJECT_REGRESSION' ;
                      neptune-ml:input ?input ;
                      neptune-ml:predicate <http://www.example.org/accountbalance> ;
                      neptune-ml:output ?output .
  }
}
```

Exemplo de previsão de objetos do SPARQL

A previsão de objetos prevê o valor do objeto para um assunto e um predicado específicos.

A seguinte consulta de previsão de objetos busca prever de qual filme a entrada do tipo `foaf:Person` gostaria:

```
?x a foaf:Person .
?x   <http://www.example.org/likes> ?m .
?m a <http://www.example.org/movie> .

## Query
SELECT * WHERE { ?input a foaf:Person .
  SERVICE neptune-ml:inference {
    neptune-ml:config neptune-ml:modelType 'OBJECT_PREDICTION' ;
                      neptune-ml:input ?input ;
```

```
                      neptune-ml:predicate <http://www.example.org/likes> ;
                      neptune-ml:output ?output ;
                      neptune-ml:outputClass <http://www.example.org/movie> .
  }
}
```

A consulta em si pode ser personalizada da seguinte forma:

```
SELECT * WHERE { ?input a foaf:Person .
  SERVICE neptune-ml:inference {
    neptune-ml:config neptune-ml:endpoint 'node-prediction-user-movie-prediction-
endpoint' ;
                      neptune-ml:iamRoleArn 'arn:aws:iam::0123456789:role/sagemaker-
role' ;

                      neptune-ml:limit "5"^^xsd:integer ;
                      neptune-ml:batchSize "40"^^xsd:integer ;
                      neptune-ml:threshold "0.1"^^xsd:double ;
                      neptune-ml:timeout "1000"^^xsd:integer ;
                      neptune-ml:outputScore ?score ;

                      neptune-ml:modelType 'OBJECT_PREDICTION' ;
                      neptune-ml:input ?input ;
                      neptune-ml:predicate <http://www.example.org/likes> ;
                      neptune-ml:output ?output ;
                      neptune-ml:outputClass <http://www.example.org/movie> .
  }
}
```

Exemplo de previsão de assuntos do SPARQL

A previsão de assuntos prevê o assunto considerando-se um predicado e um objeto.

Por exemplo, a seguinte consulta prevê quem (do tipo `foaf:User`) assistirá a determinado filme:

```
SELECT * WHERE { ?input (a foaf:Movie) .
    SERVICE neptune-ml:inference {
        neptune-ml:config neptune-ml:modelType 'SUBJECT_PREDICTION' ;
                          neptune-ml:input ?input ;
                          neptune-ml:predicate <http://aws.amazon.com/neptune/csv2rdf/
object_Property/rated> ;
                          neptune-ml:output ?output ;
```

```
                        neptune-ml:outputClass <http://aws.amazon.com/neptune/
csv2rdf/class/User> ;       }
}
```

Lista de exceções para consultas de inferência do SPARQL no Neptune ML

- **BadRequestException**: mensagem: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference expects at least 1 value for the parameter *(parameter name)*, found zero.
- **BadRequestException**: mensagem: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference expects at most 1 value for the parameter *(parameter name)*, found *(a number)* values.
- **BadRequestException**: mensagem: Invalid predicate *(predicate name)* provided for external service http://aws.amazon.com/neptune/vocab/v01/services/ml#inference query.
- **BadRequestException**: mensagem: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference expects the predicate *(predicate name)* to be defined.
- **BadRequestException**: mensagem: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference expects the value of (parameter) *(parameter name)* to be a variable, found: *(type)*"
- **BadRequestException**: mensagem: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference expects the input *(parameter name)* to be a constant, found: *(type)*.
- **BadRequestException**: mensagem: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference is expected to return only 1 value.
- **BadRequestException**: mensagem: "The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference only allows StatementPatternNodes.
- **BadRequestException**: mensagem: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference does not allow the predicate *(predicate name)*.
- **BadRequestException**: mensagem: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference predicates cannot be variables, found: *(type)*.

- **BadRequestException**: mensagem: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference predicates are expected to be part of the namespace *(namespace name)*, found: *(namespace name)*.

Referência da API de gerenciamento do Neptune ML

Sumário

Processamento de dados usando o comando **`dataprocessing`**

Use o comando `dataprocessing` do Neptune ML para criar um trabalho de processamento de dados, conferir o status, interrompê-lo ou listar todos os trabalhos ativos de processamento de dados.

Criar um trabalho de processamento de dados usando o comando **`dataprocessing`** do Neptune ML

Um comando `dataprocessing` típico do Neptune ML para criar um trabalho tem a seguinte aparência:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/dataprocessing \
  -H 'Content-Type: application/json' \
  -d '{
        "inputDataS3Location" : "s3://(Amazon S3 bucket name)/(path to your input
 folder)",
        "id" : "(a job ID for the new job)",
        "processedDataS3Location" : "s3://(S3 bucket name)/(path to your output
 folder)"
      }'
```

Um comando para iniciar o reprocessamento incremental tem a seguinte aparência:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/dataprocessing \
  -H 'Content-Type: application/json' \
  -d '{
        "inputDataS3Location" : "s3://(Amazon S3 bucket name)/(path to your input
 folder)",
        "id" : "(a job ID for this job)",
        "processedDataS3Location" : "s3://(S3 bucket name)/(path to your output
 folder)"
        "previousDataProcessingJobId" : "(the job ID of a previously completed job to
 update)"
}'
```

Parâmetros para criação de trabalhos **`dataprocessing`**

- **`id`**: (opcional) um identificador exclusivo do novo trabalho.

Tipo: string. Padrão: um UUID gerado automaticamente.

- **`previousDataProcessingJobId`**: (opcional) o ID de um trabalho de processamento de dados concluído executado em uma versão anterior dos dados.

  Tipo: string. Padrão: nenhum.

  Observação: use para processamento incremental de dados, para atualizar o modelo quando os dados do grafo forem alterados (mas não quando os dados forem excluídos).

- **`inputDataS3Location`**: (obrigatório) o URI do local do Amazon S3 em que você deseja que o SageMaker baixe os dados necessários para executar o trabalho de processamento de dados.

  Tipo: string.

- **`processedDataS3Location`**: (obrigatório) o URI do local do Amazon S3 onde você deseja que o SageMaker salve os resultados do trabalho de processamento de dados.

  Tipo: string.

- **`sagemakerIamRoleArn`**: (opcional) o ARN de um perfil do IAM para execução do SageMaker.

  Tipo: string. Observação: deve estar listado no grupo de parâmetros do cluster de banco de dados ou ocorrerá um erro.

- **`neptuneIamRoleArn`**: (opcional) o nome do recurso da Amazon (ARN) de um perfil do IAM que o SageMaker pode assumir para realizar tarefas em seu nome.

  Tipo: string. Observação: deve estar listado no grupo de parâmetros do cluster de banco de dados ou ocorrerá um erro.

- **`processingInstanceType`**: (opcional) o tipo de instância de ML usada durante o processamento de dados. A memória deve ser grande o suficiente para armazenar o conjunto de dados processado.

  Tipo: string. Padrão: o menor tipo `ml.r5` cuja memória é dez vezes maior que o tamanho dos dados de grafos exportados no disco.

  Observação: o Neptune ML pode selecionar o tipo de instância automaticamente. Consulte Selecionar uma instância para processamento de dados.

- **`processingInstanceVolumeSizeInGB`**: (opcional) o tamanho do volume do disco da instância de processamento. Tanto os dados de entrada quanto os dados processados são armazenados

em disco, portanto, o tamanho do volume deve ser grande o suficiente para conter os dois conjuntos de dados.

Tipo: número inteiro. Padrão: `0`.

Observação: se não for especificado ou for 0, o Neptune ML escolherá o tamanho do volume automaticamente com base no tamanho dos dados.

- **`processingTimeOutInSeconds`**: (opcional) tempo limite em segundos do trabalho de processamento de dados.

  Tipo: número inteiro. Padrão: `86,400` (um dia).

- **`modelType`**: (opcional) um dos dois tipos de modelo que o Neptune ML aceita no momento: modelos de grafos heterogêneos (`heterogeneous`) e grafo de conhecimento (`kge`).

  Tipo: string. Padrão: nenhum.

  Observação: se não for especificado, o Neptune ML escolherá o tipo de modelo automaticamente com base nos dados.

- **`configFileName`**: (opcional) um arquivo de especificação de dados que descreve como carregar os dados de grafos exportados para treinamento. O arquivo é gerado automaticamente pelo kit de ferramentas de exportação do Neptune.

  Tipo: string. Padrão: `training-data-configuration.json`.

- **`subnets`**: (opcional) os IDs das sub-redes na VPC do Neptune.

  Tipo: lista de strings. Padrão: nenhum.

- **`securityGroupIds`**: (opcional) os IDs do grupo de segurança da VPC.

  Tipo: lista de strings. Padrão: nenhum.

- **`volumeEncryptionKMSKey`**: (opcional) a chave AWS Key Management Service (AWS KMS) que o SageMaker usa para criptografar dados no volume de armazenamento anexado às instâncias de computação de ML que executam o trabalho de processamento.

  Tipo: string. Padrão: nenhum.

- **`enableInterContainerTrafficEncryption`**: (opcional) habilite ou desabilite a criptografia de tráfego entre contêineres em trabalhos de treinamento ou ajuste de hiperparâmetros.

  Tipo: booliano. Padrão: verdadeiro.

> **Note**
>
> O parâmetro `enableInterContainerTrafficEncryption` só está disponível na versão 1.2.0.2.R3 do mecanismo.

- **`s3OutputEncryptionKMSKey`**: (opcional) a chave AWS Key Management Service (AWS KMS) que o SageMaker usa para criptografar a saída do trabalho de treinamento.

  Tipo: string. Padrão: nenhum.

Obter o status de um trabalho de processamento de dados usando o comando **`dataprocessing`** do Neptune ML

Um exemplo de comando `dataprocessing` do Neptune ML para o status de um trabalho tem a seguinte aparência:

```
curl -s \
  "https://(your Neptune endpoint)/ml/dataprocessing/(the job ID)" \
  | python -m json.tool
```

Parâmetros para o status do trabalho **`dataprocessing`**

- **`id`**: (obrigatório) o identificador exclusivo do trabalho de processamento de dados.

  Tipo: string.
- **`neptuneIamRoleArn`**: (opcional) o ARN de um perfil do IAM que fornece ao Neptune acesso aos recursos do SageMaker e do Amazon S3.

  Tipo: string. Observação: deve estar listado no grupo de parâmetros do cluster de banco de dados ou ocorrerá um erro.

Interromper um trabalho de processamento de dados usando o comando **`dataprocessing`** do Neptune ML

Um exemplo de comando `dataprocessing` do Neptune ML para interromper um trabalho tem a seguinte aparência:

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/dataprocessing/(the job ID)"
```

Ou esta:

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/dataprocessing/(the job ID)?clean=true"
```

Parâmetros para um trabalho de interrupção **`dataprocessing`**

- **`id`**: (obrigatório) o identificador exclusivo do trabalho de processamento de dados.

  Tipo: string.
- **`neptuneIamRoleArn`**: (opcional) o ARN de um perfil do IAM que fornece ao Neptune acesso aos recursos do SageMaker e do Amazon S3.

  Tipo: string. Observação: deve estar listado no grupo de parâmetros do cluster de banco de dados ou ocorrerá um erro.
- **`clean`**: (opcional) esse sinalizador especifica que todos os artefatos do Amazon S3 devem ser excluídos quando o trabalho é interrompido.

  Tipo: booliano. Padrão: `FALSE`.

Listar trabalhos de processamento de dados ativos usando o comando **`dataprocessing`** do Neptune ML

Um exemplo de comando `dataprocessing` do Neptune ML para listar trabalhos ativos tem a seguinte aparência:

```
curl -s "https://(your Neptune endpoint)/ml/dataprocessing"
```

Ou esta:

```
curl -s "https://(your Neptune endpoint)/ml/dataprocessing?maxItems=3"
```

Parâmetros para trabalhos de lista **`dataprocessing`**

- **`maxItems`**: (opcional) o número máximo de itens a serem gerados.

Tipo: número inteiro. Padrão: `10`. Valor máximo permitido: `1024`.

- **`neptuneIamRoleArn`**: (opcional) o ARN de um perfil do IAM que fornece ao Neptune acesso aos recursos do SageMaker e do Amazon S3.

  Tipo: string. Observação: deve estar listado no grupo de parâmetros do cluster de banco de dados ou ocorrerá um erro.

Treinamento de modelos usando o comando **`modeltraining`**

Use o comando `modeltraining` do Neptune ML para criar um trabalho de treinamento de modelos, conferir o status, interrompê-lo ou listar todos os trabalhos ativos de treinamento de modelos.

Criar um trabalho de treinamento de modelos usando o comando **`modeltraining`** do Neptune ML

Um comando `modeltraining` do Neptune ML para criar um trabalho tem a seguinte aparência:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltraining
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer"
      }'
```

Um comando `modeltraining` do Neptune ML para criar um trabalho de atualização para treinamento incremental de modelos tem a seguinte aparência:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltraining
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer"
        "previousModelTrainingJobId" : "(the job ID of a completed model-training job
 to update)",
      }'
```

Um comando `modeltraining` do Neptune ML para criar um trabalho com a implementação de modelos personalizados fornecidos pelo usuário tem a seguinte aparência:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltraining
```

```
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer"
        "modelName": "custom",
        "customModelTrainingParameters" : {
          "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your Python
 module)",
          "trainingEntryPointScript": "(your training script entry-point name in the
 Python module)",
          "transformEntryPointScript": "(your transform script entry-point name in the
 Python module)"
        }
      }'
```

Parâmetros para criação de trabalhos **`modeltraining`**

- **`id`**: (opcional) um identificador exclusivo do novo trabalho.

  Tipo: string Padrão: um UUID gerado automaticamente.
- **`dataProcessingJobId`**: (obrigatório) o ID do trabalho de processamento de dados concluído que criou os dados com os quais o treinamento funcionará.

  Tipo: string
- **`trainModelS3Location`**: (obrigatório) o local no Amazon S3 onde os artefatos do modelo devem ser armazenados.

  Tipo: string
- **`previousModelTrainingJobId`**: (opcional) o ID de um trabalho de treinamento de modelos concluído que você deseja atualizar de modo incremental com base nos dados atualizados.

  Tipo: string Padrão: nenhum.
- **`sagemakerIamRoleArn`**: (opcional) o ARN de um perfil do IAM para execução do SageMaker.

  Tipo: string Observação: deve estar listado no grupo de parâmetros do cluster de banco de dados ou ocorrerá um erro.
- **`neptuneIamRoleArn`**: (opcional) o ARN de um perfil do IAM que fornece ao Neptune acesso aos recursos do SageMaker e do Amazon S3.

Tipo: string Observação: deve estar listado no grupo de parâmetros do cluster de banco de dados ou ocorrerá um erro.

- **`modelName`**: (opcional) o tipo de modelo para treinamento. Por padrão, o modelo de ML é automaticamente baseado no `modelType` usado no processamento de dados, mas você pode especificar um tipo de modelo diferente aqui.

  Tipo: string Padrão: `rgcn` para grafos heterogêneos e `kge` para grafos de conhecimento. Valores válidos: para grafos heterogêneos: `rgcn`. Para grafos `kge`:`transe`, `distmult` ou `rotate`. Para uma implementação de modelos personalizados: `custom`.

- **`baseProcessingInstanceType`**: (opcional) o tipo de instância de ML usada na preparação e gerenciamento do treinamento de modelos de ML.

  Tipo: string Observação: é uma instância de CPU escolhida com base nos requisitos de memória para processar os dados e o modelo de treinamento. Consulte Selecionar uma instância para treinamento e transformação de modelos.

- **`trainingInstanceType`**: (opcional) o tipo de instância de ML usada para treinamento de modelos. Todos os modelos do Neptune ML são compatíveis com o treinamento de CPU, GPU e multiGPU.

  Tipo: string Padrão: `ml.p3.2xlarge`.

  Observação: a escolha do tipo de instância certo para treinamento depende do tipo de tarefa, do tamanho do grafo e do orçamento. Consulte Selecionar uma instância para treinamento e transformação de modelos.

- **`trainingInstanceVolumeSizeInGB`**: (opcional) o tamanho do volume do disco da instância de treinamento. Tanto os dados de entrada quanto o modelo de saída são armazenados em disco, portanto, o tamanho do volume deve ser grande o suficiente para conter os dois conjuntos de dados.

  Tipo: número inteiro. Padrão: `0`.

  Observação: se não for especificado ou for 0, o Neptune ML selecionará um tamanho de volume de disco com base na recomendação gerada na etapa de processamento de dados. Consulte Selecionar uma instância para treinamento e transformação de modelos.

- **`trainingTimeOutInSeconds`**: (opcional) tempo limite em segundos para o trabalho de treinamento.

Tipo: número inteiro. Padrão: `86,400` (um dia).

- **`maxHPONumberOfTrainingJobs`**: número total máximo de trabalhos de treinamento a serem iniciados para o trabalho de ajuste de hiperparâmetros.

  Tipo: número inteiro. Padrão: 2.

  Observação: o Neptune ML ajusta automaticamente os hiperparâmetros do modelo de machine learning. Para obter um modelo com bom desempenho, use pelo menos dez trabalhos (em outras palavras, defina `maxHPONumberOfTrainingJobs` como dez). Em geral, quanto mais ajustes forem executados, melhores serão os resultados.

- **`maxHPOParallelTrainingJobs`**: número máximo de trabalhos de treinamento paralelos a serem iniciados para o trabalho de ajuste de hiperparâmetros.

  Tipo: número inteiro. Padrão: 2.

  Observação: o número de trabalhos paralelos que você pode executar é limitado pelos recursos disponíveis na instância de treinamento.

- **`subnets`**: (opcional) os IDs das sub-redes na VPC do Neptune.

  Tipo: lista de strings. Padrão: nenhum.

- **`securityGroupIds`**: (opcional) os IDs do grupo de segurança da VPC.

  Tipo: lista de strings. Padrão: nenhum.

- **`volumeEncryptionKMSKey`**: (opcional) a chave AWS Key Management Service (AWS KMS) que o SageMaker usa para criptografar dados no volume de armazenamento anexado às instâncias de computação de ML que executam o trabalho de treinamento.

  Tipo: string Padrão: nenhum.

- **`s3OutputEncryptionKMSKey`**: (opcional) a chave AWS Key Management Service (AWS KMS) que o SageMaker usa para criptografar a saída do trabalho de processamento.

  Tipo: string Padrão: nenhum.

- **`enableInterContainerTrafficEncryption`**: (opcional) habilite ou desabilite a criptografia de tráfego entre contêineres em trabalhos de treinamento ou ajuste de hiperparâmetros.

  Tipo: booliano. Padrão: verdadeiro.

> **Note**
>
> O parâmetro `enableInterContainerTrafficEncryption` só está disponível na versão 1.2.0.2.R3 do mecanismo.

- **`enableManagedSpotTraining`**: (opcional) otimiza o custo de treinamento de modelos de machine learning usando instâncias spot do Amazon Elastic Compute Cloud. Para obter mais informações, consulte Managed Spot Training in Amazon SageMaker.

  Tipo: booliano. Padrão: falso.

- **`customModelTrainingParameters`**: (opcional) a configuração para treinamento de modelos personalizados. Trata-se de um objeto JSON com os seguintes campos:

  - **`sourceS3DirectoryPath`**: (obrigatório) o caminho para o local do Amazon S3 onde o módulo Python que implementa seu modelo está localizado. Isso deve apontar para uma localização válida existente do Amazon S3 que contenha, no mínimo, um script de treinamento, um script de transformação e um arquivo `model-hpo-configuration.json`.
  - **`trainingEntryPointScript`**: (opcional) o nome do ponto de entrada no módulo de um script que executa o treinamento de modelos e usa hiperparâmetros como argumentos de linha de comando, incluindo hiperparâmetros fixos.

    Padrão: `training.py`.
  - **`transformEntryPointScript`**: (opcional) o nome do ponto de entrada no módulo de um script que deve ser executado após a identificação do melhor modelo da pesquisa de hiperparâmetros, para calcular os artefatos do modelo necessários para a implantação do modelo. Ele deve ser capaz de ser executado sem argumentos de linha de comando.

    Padrão: `transform.py`.

- **`maxWaitTime`**: (opcional) o tempo máximo de espera, em segundos, ao realizar o treinamento de modelos usando instâncias spot. Deve ser maior que `trainingTimeOutInSeconds`.

  Tipo: número inteiro.

Obter o status de um trabalho de treinamento de modelos usando o comando **`modeltraining`** do Neptune ML

Um exemplo de comando `modeltraining` do Neptune ML para o status de um trabalho tem a seguinte aparência:

```
curl -s \
  "https://(your Neptune endpoint)/ml/modeltraining/(the job ID)" \
  | python -m json.tool
```

Parâmetros para o status do trabalho **`modeltraining`**

- **`id`**: (obrigatório) o identificador exclusivo do trabalho de treinamento de modelos.

  Tipo: string
- **`neptuneIamRoleArn`**: (opcional) o ARN de um perfil do IAM que fornece ao Neptune acesso aos recursos do SageMaker e do Amazon S3.

  Tipo: string Observação: deve estar listado no grupo de parâmetros do cluster de banco de dados ou ocorrerá um erro.

Interromper um trabalho de treinamento de modelos usando o comando **`modeltraining`** do Neptune ML

Um exemplo de comando `modeltraining` do Neptune ML para interromper um trabalho tem a seguinte aparência:

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/modeltraining/(the job ID)"
```

Ou esta:

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/modeltraining/(the job ID)?clean=true"
```

Parâmetros para um trabalho de interrupção **`modeltraining`**

- **`id`**: (obrigatório) o identificador exclusivo do trabalho de treinamento de modelos.

Tipo: string

- **`neptuneIamRoleArn`**: (opcional) o ARN de um perfil do IAM que fornece ao Neptune acesso aos recursos do SageMaker e do Amazon S3.

  Tipo: string Observação: deve estar listado no grupo de parâmetros do cluster de banco de dados ou ocorrerá um erro.
- **`clean`**: (opcional) esse sinalizador especifica que todos os artefatos do Amazon S3 devem ser excluídos quando o trabalho é interrompido.

  Tipo: booliano. Padrão: `FALSE`.

Listar trabalhos de treinamento de modelos usando o comando **`modeltraining`** do Neptune ML

Um exemplo de comando `modeltraining` do Neptune ML para listar trabalhos ativos tem a seguinte aparência:

```
curl -s "https://(your Neptune endpoint)/ml/modeltraining" | python -m json.tool
```

Ou esta:

```
curl -s "https://(your Neptune endpoint)/ml/modeltraining?maxItems=3" | python -m
 json.tool
```

Parâmetros para trabalhos de lista **`modeltraining`**

- **`maxItems`**: (opcional) o número máximo de itens a serem gerados.

  Tipo: número inteiro. Padrão: `10`. Valor máximo permitido: `1024`.
- **`neptuneIamRoleArn`**: (opcional) o ARN de um perfil do IAM que fornece ao Neptune acesso aos recursos do SageMaker e do Amazon S3.

  Tipo: string Observação: deve estar listado no grupo de parâmetros do cluster de banco de dados ou ocorrerá um erro.

Transformação de modelos usando o comando **`modeltransform`**

Use o comando `modeltransform` do Neptune ML para criar um trabalho de transformação de modelos, conferir o status, interrompê-lo ou listar todos os trabalhos ativos de transformação de modelos.

Criar um trabalho de transformação de modelos usando o comando **`modeltransform`** do Neptune ML

Um comando `modeltransform` do Neptune ML para criar um trabalho de transformação incremental, sem novo treinamento de modelos, tem a seguinte aparência:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltransform
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-transform job ID)",
        "dataProcessingJobId" : "(the job-id of a completed data-processing job)",
        "mlModelTrainingJobId" : "(the job-id of a completed model-training job)",
        "modelTransformOutputS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-
transform"
      }'
```

Um comando `modeltransform` do Neptune ML para criar um trabalho a partir de um trabalho de treinamento concluído do SageMaker tem a seguinte aparência:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltransform
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-transform job ID)",
        "trainingJobName" : "(name of a completed SageMaker training job)",
        "modelTransformOutputS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-
transform",
        "baseProcessingInstanceType" : ""
      }'
```

Um comando `modeltransform` do Neptune ML para criar um trabalho que use uma implementação de modelos personalizados tem a seguinte aparência:

```
curl \
```

```
  -X POST https://(your Neptune endpoint)/ml/modeltransform
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "trainingJobName" : "(name of a completed SageMaker training job)",
        "modelTransformOutputS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-
transform/"
        "customModelTransformParameters" : {
          "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your Python
 module)",
          "transformEntryPointScript": "(your transform script entry-point name in the
 Python module)"
        }
      }'
```

Parâmetros para criação de trabalhos **`modeltransform`**

- **`id`**: (opcional) um identificador exclusivo do novo trabalho.

  Tipo: string. Padrão: um UUID gerado automaticamente.
- **`dataProcessingJobId`**: o ID de um trabalho de processamento de dados concluído.

  Tipo: string.

  Observação: você deve incluir `dataProcessingJobId`, `mlModelTrainingJobId`, ou `trainingJobName`.
- **`mlModelTrainingJobId`**: o ID de um trabalho de treinamento de modelos concluído.

  Tipo: string.

  Observação: você deve incluir `dataProcessingJobId`, `mlModelTrainingJobId` ou `trainingJobName`.
- **`trainingJobName`**: o nome de um trabalho de treinamento concluído do SageMaker.

  Tipo: string.

  Observação: você deve incluir os parâmetros `dataProcessingJobId` e `mlModelTrainingJobId` ou `trainingJobName`.
- **`sagemakerIamRoleArn`**: (opcional) o ARN de um perfil do IAM para execução do SageMaker.

Tipo: string. Observação: deve estar listado no grupo de parâmetros do cluster de banco de dados ou ocorrerá um erro.

- **`neptuneIamRoleArn`**: (opcional) o ARN de um perfil do IAM que fornece ao Neptune acesso aos recursos do SageMaker e do Amazon S3.

  Tipo: string. Observação: deve estar listado no grupo de parâmetros do cluster de banco de dados ou ocorrerá um erro.

- **`customModelTransformParameters`** : (opcional) informações de configuração para uma transformação de modelos usando um modelo personalizado. O objeto `customModelTransformParameters` contém os seguintes campos, que devem ter valores compatíveis com os parâmetros do modelo salvos do trabalho de treinamento:

  - **`sourceS3DirectoryPath`**: (obrigatório) o caminho para o local do Amazon S3 onde o módulo Python que implementa seu modelo está localizado. Isso deve apontar para uma localização válida existente do Amazon S3 que contenha, no mínimo, um script de treinamento, um script de transformação e um arquivo `model-hpo-configuration.json`.
  - **`transformEntryPointScript`**: (opcional) o nome do ponto de entrada no módulo de um script que deve ser executado após a identificação do melhor modelo da pesquisa de hiperparâmetros, para calcular os artefatos do modelo necessários para a implantação do modelo. Ele deve ser capaz de ser executado sem argumentos de linha de comando.

    Padrão: `transform.py`.

- **`baseProcessingInstanceType`**: (opcional) o tipo de instância de ML usada na preparação e gerenciamento do treinamento de modelos de ML.

  Tipo: string. Observação: é uma instância de CPU escolhida com base nos requisitos de memória para processar os dados e o modelo de transformação. Consulte Selecionar uma instância para treinamento e transformação de modelos.

- **`baseProcessingInstanceVolumeSizeInGB`**: (opcional) o tamanho do volume do disco da instância de treinamento. Tanto os dados de entrada quanto o modelo de saída são armazenados em disco, portanto, o tamanho do volume deve ser grande o suficiente para conter os dois conjuntos de dados.

  Tipo: número inteiro. Padrão: `0`.

Observação: se não for especificado ou for 0, o Neptune ML selecionará um tamanho de volume de disco com base na recomendação gerada na etapa de processamento de dados. Consulte Selecionar uma instância para treinamento e transformação de modelos.

- **`subnets`**: (opcional) os IDs das sub-redes na VPC do Neptune.

  Tipo: lista de strings. Padrão: nenhum.

- **`securityGroupIds`**: (opcional) os IDs do grupo de segurança da VPC.

  Tipo: lista de strings. Padrão: nenhum.

- **`volumeEncryptionKMSKey`**: (opcional) a chave AWS Key Management Service (AWS KMS) que o SageMaker usa para criptografar dados no volume de armazenamento anexado às instâncias de computação de ML que executam o trabalho de transformação.

  Tipo: string. Padrão: nenhum.

- **`enableInterContainerTrafficEncryption`**: (opcional) habilite ou desabilite a criptografia de tráfego entre contêineres em trabalhos de treinamento ou ajuste de hiperparâmetros.

  Tipo: booliano. Padrão: verdadeiro.

  > **Note**
  >
  > O parâmetro `enableInterContainerTrafficEncryption` só está disponível na versão 1.2.0.2.R3 do mecanismo.

- **`s3OutputEncryptionKMSKey`**: (opcional) a chave AWS Key Management Service (AWS KMS) que o SageMaker usa para criptografar a saída do trabalho de processamento.

  Tipo: string. Padrão: nenhum.

Obter o status de um trabalho de transformação de modelos usando o comando **`modeltransform`** do Neptune ML

Um exemplo de comando `modeltransform` do Neptune ML para o status de um trabalho tem a seguinte aparência:

```
curl -s \
  "https://(your Neptune endpoint)/ml/modeltransform/(the job ID)" \
```

```
  | python -m json.tool
```

Parâmetros para o status do trabalho **`modeltransform`**

- **`id`**: (obrigatório) o identificador exclusivo do trabalho de transformação de modelos.

  Tipo: string.
- **`neptuneIamRoleArn`**: (opcional) o ARN de um perfil do IAM que fornece ao Neptune acesso aos recursos do SageMaker e do Amazon S3.

  Tipo: string. Observação: deve estar listado no grupo de parâmetros do cluster de banco de dados ou ocorrerá um erro.

Interromper um trabalho de transformação de modelos usando o comando **`modeltransform`** do Neptune ML

Um exemplo de comando `modeltransform` do Neptune ML para interromper um trabalho tem a seguinte aparência:

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/modeltransform/(the job ID)"
```

Ou esta:

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/modeltransform/(the job ID)?clean=true"
```

Parâmetros para um trabalho de interrupção **`modeltransform`**

- **`id`**: (obrigatório) o identificador exclusivo do trabalho de transformação de modelos.

  Tipo: string.
- **`neptuneIamRoleArn`**: (opcional) o ARN de um perfil do IAM que fornece ao Neptune acesso aos recursos do SageMaker e do Amazon S3.

  Tipo: string. Observação: deve estar listado no grupo de parâmetros do cluster de banco de dados ou ocorrerá um erro.
- **`clean`**: (opcional) esse sinalizador especifica que todos os artefatos do Amazon S3 devem ser excluídos quando o trabalho é interrompido.

Tipo: booliano. Padrão: `FALSE`.

Listar trabalhos ativos de transformação de modelos usando o comando **`modeltransform`** do Neptune ML

Um exemplo de comando `modeltransform` do Neptune ML para listar trabalhos ativos tem a seguinte aparência:

```
curl -s "https://(your Neptune endpoint)/ml/modeltransform" | python -m json.tool
```

Ou esta:

```
curl -s "https://(your Neptune endpoint)/ml/modeltransform?maxItems=3" | python -m
 json.tool
```

Parâmetros para trabalhos de lista **`modeltransform`**

- **`maxItems`**: (opcional) o número máximo de itens a serem gerados.

  Tipo: número inteiro. Padrão: `10`. Valor máximo permitido: `1024`.
- **`neptuneIamRoleArn`**: (opcional) o ARN de um perfil do IAM que fornece ao Neptune acesso aos recursos do SageMaker e do Amazon S3.

  Tipo: string. Observação: deve estar listado no grupo de parâmetros do cluster de banco de dados ou ocorrerá um erro.

Gerenciar endpoints de inferência usando o comando **endpoints**

Use o comando `endpoints` do Neptune ML para criar um endpoint de inferência, conferir o status, excluí-lo ou listar endpoints de inferência existentes.

Criar um endpoint de inferência usando o comando **endpoints** do Neptune ML

Um comando `endpoints` do Neptune ML para criar um endpoint de inferência a partir de um modelo criado por um trabalho de treinamento tem a seguinte aparência:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/endpoints
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "mlModelTrainingJobId": "(the model-training job-id of a completed job)"
      }'
```

Um comando `endpoints` do Neptune ML para atualizar um endpoint de inferência existente a partir de um modelo criado por um trabalho de treinamento tem a seguinte aparência:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/endpoints
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "update" : "true",
        "mlModelTrainingJobId": "(the model-training job-id of a completed job)"
      }'
```

Um comando `endpoints` do Neptune ML para criar um endpoint de inferência a partir de um modelo criado por um trabalho de transformação de modelos tem a seguinte aparência:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/endpoints
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "mlModelTransformJobId": "(the model-training job-id of a completed job)"
      }'
```

Um comando `endpoints` do Neptune ML para atualizar um endpoint de inferência existente a partir de um modelo criado por um trabalho de transformação de modelos tem a seguinte aparência:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/endpoints
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "update" : "true",
        "mlModelTransformJobId": "(the model-training job-id of a completed job)"
      }'
```

Parâmetros para criação de endpoints de inferência **`endpoints`**

- **`id`**: (opcional) um identificador exclusivo para o novo endpoint de inferência.

  Tipo: string. Padrão: um nome com carimbo de data e hora gerado automaticamente.
- **`mlModelTrainingJobId`**: o ID do trabalho de treinamento de modelos concluído que criou o modelo para o qual o endpoint de inferência apontará.

  Tipo: string.

  Observação: é necessário fornecer o `mlModelTrainingJobId` ou o `mlModelTransformJobId`.
- **`mlModelTransformJobId`**: o ID do trabalho de transformação de modelos concluído.

  Tipo: string.

  Observação: é necessário fornecer o `mlModelTrainingJobId` ou o `mlModelTransformJobId`.
- **`update`**: (opcional) se presente, esse parâmetro indica que se trata de uma solicitação de atualização.

  Tipo: booliano. Padrão: `false`

  Observação: é necessário fornecer o `mlModelTrainingJobId` ou o `mlModelTransformJobId`.
- **`neptuneIamRoleArn`**: (opcional) o ARN de um perfil do IAM que fornece ao Neptune acesso aos recursos do SageMaker e do Amazon S3.

Tipo: string. Observação: deve estar listado no grupo de parâmetros do cluster de banco de dados ou será lançado um erro.

- **`modelName`**: (opcional) o tipo de modelo para treinamento. Por padrão, o modelo de ML é automaticamente baseado no `modelType` usado no processamento de dados, mas você pode especificar um tipo de modelo diferente aqui.

  Tipo: string. Padrão: `rgcn` para grafos heterogêneos e `kge` para grafos de conhecimento. Valores válidos: para grafos heterogêneos: `rgcn`. Para grafos de conhecimento:`kge`, `transe`, `distmult` ou `rotate`.

- **`instanceType`**: (opcional) o tipo de instância de ML usada para serviços on-line.

  Tipo: string. Padrão: `ml.m5.xlarge`.

  Observação: selecionar a instância de ML para um endpoint de inferência depende do tipo de tarefa, do tamanho do grafo e do orçamento. Consulte Selecionar uma instância para um endpoint de inferência.

- **`instanceCount`**: (opcional) o número mínimo de instâncias do Amazon EC2 a serem implantadas em um endpoint para previsão.

  Tipo: número inteiro. Padrão: 1.

- **`volumeEncryptionKMSKey`**: (opcional) a chave AWS Key Management Service (AWS KMS) que o SageMaker usa para criptografar dados no volume de armazenamento anexado às instâncias de computação de ML que executam os endpoints.

  Tipo: string. Padrão: nenhum.

Obter o status de um endpoint de inferência usando o comando **`endpoints`** do Neptune ML

Um exemplo de comando `endpoints` do Neptune ML para o status de um endpoint de instância tem a seguinte aparência:

```
curl -s \
  "https://(your Neptune endpoint)/ml/endpoints/(the inference endpoint ID)" \
  | python -m json.tool
```

Parâmetros para o status do endpoint da instância **`endpoints`**

- **`id`**: (obrigatório) o identificador exclusivo do endpoint de inferência.

  Tipo: string.
- **`neptuneIamRoleArn`**: (opcional) o ARN de um perfil do IAM que fornece ao Neptune acesso aos recursos do SageMaker e do Amazon S3.

  Tipo: string. Observação: deve estar listado no grupo de parâmetros do cluster de banco de dados ou será lançado um erro.

Excluir um endpoint de instância usando o comando **`endpoints`** do Neptune ML

Um exemplo de comando `endpoints` do Neptune ML para exclusão de um endpoint de instância tem a seguinte aparência:

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/endpoints/(the inference endpoint ID)"
```

Ou esta:

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/endpoints/(the inference endpoint ID)?
clean=true"
```

Parâmetros para exclusão **`endpoints`** de um endpoint de inferência

- **`id`**: (obrigatório) o identificador exclusivo do endpoint de inferência.

  Tipo: string.
- **`neptuneIamRoleArn`**: (opcional) o ARN de um perfil do IAM que fornece ao Neptune acesso aos recursos do SageMaker e do Amazon S3.

  Tipo: string. Observação: deve estar listado no grupo de parâmetros do cluster de banco de dados ou será lançado um erro.
- **`clean`**: (opcional) indica que todos os artefatos relacionados a esse endpoint também devem ser excluídos.

  Tipo: booliano. Padrão: `FALSE`.

Listar endpoints de inferência usando o comando **`endpoints`** do Neptune ML

Um comando `endpoints` do Neptune ML para listar endpoints de inferência tem a seguinte aparência:

```
curl -s "https://(your Neptune endpoint)/ml/endpoints" \
  | python -m json.tool
```

Ou esta:

```
curl -s "https://(your Neptune endpoint)/ml/endpoints?maxItems=3" \
  | python -m json.tool
```

Parâmetros para listar endpoints de inferência **`dataprocessing`**

- **`maxItems`**: (opcional) o número máximo de itens a serem gerados.

  Tipo: número inteiro. Padrão: `10`. Valor máximo permitido: `1024`.
- **`neptuneIamRoleArn`**: (opcional) o ARN de um perfil do IAM que fornece ao Neptune acesso aos recursos do SageMaker e do Amazon S3.

  Tipo: string. Observação: deve estar listado no grupo de parâmetros do cluster de banco de dados ou será lançado um erro.

Erros e exceções da API de gerenciamento do Neptune ML

Todas as exceções da API de gerenciamento do Neptune ML exibem um código HTTP 400. Depois de receber qualquer uma dessas exceções, o comando que gerou a exceção não deve ser repetido.

- **`MissingParameterException`**: mensagem de erro:

  ```
  Required credentials are missing. Please add IAM role to the cluster or pass as a parameter to this request.
  ```
- **`InvalidParameterException`**: mensagens de erro:
  - `Invalid ML instance type.`
  - `Invalid ID provided. ID can be 1-48 alphanumeric characters.`
  - `Invalid ID provided. Must contain only letters, digits, or hyphens.`
  - `Invalid ID provided. Please check whether a resource with the given ID exists.`
  - `Another resource with same ID already exists. Please use a new ID.`
  - `Failed to stop the job because it has already completed or failed.`
- **`BadRequestException`**: mensagens de erro:
  - `Invalid S3 URL or incorrect S3 permissions. Please check your S3 configuration.`
  - `Provided ModelTraining job has not completed.`
  - `Provided SageMaker Training job has not completed.`
  - `Provided MLDataProcessing job is not completed.`
  - `Provided MLModelTraining job doesn't exist.`
  - `Provided ModelTransformJob doesn't exist.`
  - `Unable to find SageMaker resource. Please check your input.`

Limites do Neptune ML

- Os tipos de inferência atualmente compatíveis são classificação de nós, regressão de nós, classificação de bordas, regressão de bordas e previsão de links (consulte Capacidades do Neptune ML).
- O tamanho máximo do grafo que o Neptune ML pode aceitar depende da quantidade de memória e armazenamento necessários durante a preparação dos dados, o treinamento de modelos e a inferência.
  - O tamanho máximo da memória de uma instância de processamento de dados do SageMaker é 768 GB. Como resultado, a fase de processamento de dados falhará se precisar de mais de 768 GB de memória.
  - O tamanho máximo da memória de uma instância de treinamento do SageMaker é 732 GB. Como resultado, a fase de treinamento falhará se precisar de mais de 732 GB de memória.
- O tamanho máximo de uma carga útil de inferência para um endpoint do SageMaker é 6 MiB. Como resultado, a inferência indutiva falhará se a carga útil do subgrafo exceder esse tamanho.
- No momento, o Neptune ML está disponível somente em regiões nas quais o Neptune e os outros serviços dos quais ele depende (como o AWS Lambda, o Amazon API Gateway e o Amazon SageMaker) são todos compatíveis.

  Há diferenças na China (Pequim) e na China (Ningxia) relacionadas ao uso padrão da autenticação do IAM, conforme explicado aqui além de outras diferenças.
- No momento, os endpoints de inferência de previsão de links lançados pelo Neptune ML só podem prever possíveis links com nós que estavam presentes no grafo durante o treinamento.

  Por exemplo, considere um grafo com vértices `User` e `Movie` e bordas `Rated`. Usando um modelo de recomendação de previsão de links do Neptune ML correspondente, é possível adicionar um novo usuário ao grafo e fazer com que o modelo preveja filmes para ele, mas o modelo só pode recomendar filmes que estavam presentes durante o treinamento de modelos. Embora a incorporação de nós `User` seja calculada em tempo real usando o subgrafo local e o modelo de GNN e, portanto, possa mudar com o tempo à medida que os usuários avaliam os filmes, ela é comparada às incorporações de filmes estáticas pré-calculadas para a recomendação final.
- Os modelos KGE compatíveis com o Neptune ML funcionam apenas para tarefas de previsão de links, e as representações são específicas de vértices e tipos de borda presentes no grafo durante o treinamento. Isso significa que todos os vértices e os tipos de borda mencionados em

uma consulta de inferência devem estar presentes no grafo durante o treinamento. As previsões para novos tipos de borda ou vértices não podem ser feitas sem treinar novamente o modelo.

Limitações de recursos do SageMaker

Dependendo das atividades e do uso de recursos ao longo do tempo, é possível encontrar mensagens de erro dizendo que você excedeu sua cota (ResourceLimitExceeded). É necessário aumentar a escala verticalmente dos recursos do SageMaker. Siga as etapas no procedimento Solicitar um aumento da cota de serviço para recursos do SageMaker nesta página para solicitar aumento da cota ao AWS Support.

Os nomes dos recursos do SageMaker correspondem às fases do Neptune ML da seguinte forma:

- O `ProcessingJob` do SageMaker é usado por trabalhos de processamento de dados, treinamento e transformação de modelos do Neptune.
- O `HyperParameterTuningJob` do SageMaker é usado por trabalhos de treinamento de modelos do Neptune.
- O `TrainingJob` do SageMaker é usado por trabalhos de treinamento de modelos do Neptune.
- O `Endpoint` do SageMaker é usado por endpoints de inferência do Neptune.

Monitorar recursos do Amazon Neptune

O Amazon Neptune é compatível com vários métodos para monitorar o desempenho e o uso:

- Status da instância: confira a integridade do mecanismo de banco de dados de grafos de um cluster do Neptune, descubra qual versão do mecanismo está instalada e obtenha outras informações relacionadas à instância usando a API de status da instância.
- API de resumo do grafo: a API de resumo do grafo permite que você obtenha rapidamente uma compreensão geral do tamanho e do conteúdo dos dados de grafos.

> **Note**
>
> Como a API de resumo do grafo depende das estatísticas do DFE, ela só está disponível quando as estatísticas estão habilitadas, o que não é o caso dos tipos de instância T3 e T4g.

- Amazon CloudWatch — Neptune envia métricas automaticamente e também oferece suporte CloudWatch a alarmes. CloudWatch Para ter mais informações, consulte the section called “Usando CloudWatch”.
- Arquivos de log de auditoria: visualize, baixe ou observe arquivos de log de banco de dados usando o console do Neptune. Para ter mais informações, consulte the section called “Logs de auditoria com Neptune”.
- Publicação de registros no Amazon CloudWatch Logs — Você pode configurar um cluster de banco de dados Neptune para publicar dados de log de auditoria em um grupo de logs no Amazon Logs. CloudWatch Com o CloudWatch Logs, você pode realizar análises em tempo real dos dados de log, usar CloudWatch para criar alarmes e visualizar métricas, e usar o CloudWatch Logs para armazenar seus registros de log em um armazenamento altamente durável. Consulte Registros de Netuno CloudWatch .
- AWS CloudTrail— O Neptune suporta o registro de API usando. CloudTrail Para ter mais informações, consulte the section called “Registrando chamadas da API Neptune com AWS CloudTrail”.
- Assinaturas de notificação de eventos: inscreva-se nos eventos do Neptune para se manter informado sobre o que está acontecendo. Para ter mais informações, consulte the section called “Notificações de eventos”.

- Marcação: use tags para adicionar metadados aos recursos do Neptune e monitorar o uso com base em tags. Para ter mais informações, consulte the section called “Marcar os recursos do Neptune”.

Tópicos

Conferir o status de integridade de uma instância do Neptune

O Amazon Neptune fornece um mecanismo para conferir o status do banco de dados de grafos no host. Também é uma boa maneira de confirmar que você pode se conectar a uma instância.

Para verificar a integridade de uma instância e obter o status do cluster de banco de dados usando `curl`:

```
curl -G https://your-neptune-endpoint:port/status
```

Ou, a partir da versão 1.2.1.0.R6 do mecanismo, você pode usar o seguinte comando da CLI:

```
aws neptunedata get-engine-status
```

Se a instância estiver íntegra, o comando `status` retornará um objeto JSON com os seguintes campos:

- **`status`**: defina como `"healthy"` se a instância não estiver com problemas.

Se a instância estiver se recuperando de um travamento ou sendo reinicializada e houver transações ativas em execução no desligamento do servidor mais recente, o `status` será definido como `"recovery"`.

- **`startTime`**: defina como a hora UTC em que o processo do servidor atual foi iniciado.
- **`dbEngineVersion`**: defina como a versão do mecanismo do Neptune em execução no cluster de banco de dados.

  Se essa versão do mecanismo tiver sido corrigida manualmente desde que foi lançada, o número da versão terá o prefixo `"Patch-"`.
- **`role`** defina como `"reader"` se a instância for uma réplica de leitura ou como `"writer"` se a instância for a principal.
- **`dfeQueryEngine`**: defina como `"enabled"` se o mecanismo do DFE estiver totalmente habilitado ou `viaQueryHint` se o mecanismo do DFE for usado somente com consultas que tenham a dica de consulta `useDFE` definida como `true` (`viaQueryHint` é o padrão).
- **`gremlin`**: contém informações sobre a linguagem de consulta do Gremlin disponível no cluster. Especificamente, ele contém um `version` campo que especifica a TinkerPop versão atual que está sendo usada pelo mecanismo.
- **`sparql`**: contém informações sobre a linguagem de consulta do SPARQL disponível no cluster. Especificamente, contém um campo `version` que especifica a versão atual do SPARQL que está sendo usada pelo mecanismo.
- **`opencypher`**: contém informações sobre a linguagem de consulta do openCypher disponível no cluster. Especificamente, contém um campo `version` que especifica a versão atual do openCypher que está sendo usada pelo mecanismo.
- **`labMode`**: contém as configurações Modo de laboratório usadas pelo mecanismo.
- **`rollingBackTrxCount`**: se houver transações sendo revertidas, esse campo será definido como o número dessas transações. Se não houver nenhuma, o campo não aparecerá.
- **`rollingBackTrxEarliestStartTime`**: defina como a hora de início da transação mais antiga revertida. Se nenhuma transação estiver sendo revertida, o campo não aparecerá.
- **`features`**: contém informações de status sobre os atributos habilitados no cluster de banco de dados:
  - **`lookupCache`**: o status atual do Cache de pesquisa. Esse campo aparece somente em tipos de instância `R5d`, pois essas são as únicas instâncias em que um cache de pesquisa pode existir. O campo é um objeto JSON no formato:

```
"lookupCache": {
  "status": "current lookup cache status"
}
```

Em uma instância R5d:

- Se o cache de pesquisa estiver habilitado, o status será listado como `"Available"`.
- Se o cache de pesquisa estiver desabilitado, o status será listado como `"Disabled"`.
- Se o limite de disco tiver sido atingido na instância, o status será listado como `"Read Only Mode - Storage Limit Reached"`.

- **`ResultCache`**: o status atual do Armazenar em cache os resultados da consulta. Esse campo é um objeto JSON no formato:

```
"ResultCache": {
  "status": "current results cache status"
}
```

  - Se o cache de resultados tiver sido habilitado, o status será listado como `"Available"`.
  - Se o cache estiver desabilitado, o status será listado como `"Disabled"`.
- **`IAMAuthentication`**— Especifica se a autenticação AWS Identity and Access Management (IAM) foi habilitada ou não em seu cluster de banco de dados:
  - Se a autenticação do IAM estiver habilitada, o status será listado como `"enabled"`.
  - Se a autenticação do IAM estiver desabilitada, o status será listado como `"disabled"`.
- **`Streams`**: especifica se os fluxos do Neptune foram habilitados ou não no cluster de banco de dados:
  - Se os fluxos estiverem habilitados, o status será listado como `"enabled"`.
  - Se os fluxos estiverem desabilitados, o status será listado como `"disabled"`.
- **`AuditLog`**: igual a `enabled` se os logs de auditoria estiverem habilitados. Caso contrário, `disabled`.
- **`SlowQueryLogs`**: igual a `info` ou `debug` se o registro em log de consultas lentas estiver habilitado. Caso contrário, `disabled`.
- **`QueryTimeout`**: o valor, em milissegundos, do tempo limite da consulta.
- **`settings`**: configurações aplicadas à instância:

  - **`clusterQueryTimeoutInMs`**: o valor, em milissegundos, do tempo limite da consulta, definido para todo o cluster.
  - **`SlowQueryLogsThreshold`**: o valor, em milissegundos, do tempo limite da consulta, definido para todo o cluster.
- **`serverlessConfiguration`**: configurações sem servidor para um cluster se ele estiver sendo executado sem servidor:
  - **`minCapacity`**: o menor tamanho para o qual uma instância sem servidor no cluster de banco de dados pode ser reduzida, em unidades de capacidade do Neptune (NCUs).
  - **`maxCapacity`**: o maior tamanho para o qual uma instância sem servidor no cluster de banco de dados pode ser aumentada, em unidades de capacidade do Neptune (NCUs).

Exemplo da saída do comando instance status

Veja um exemplo da saída do comando instance status (nesse caso, executado em uma instância R5d):

```
{
  'status': 'healthy',
  'startTime': 'Thu Aug 24 21:47:12 UTC 2023',
  'dbEngineVersion': '1.2.1.0.R4',
  'role': 'writer',
  'dfeQueryEngine': 'viaQueryHint',
  'gremlin': {'version': 'tinkerpop-3.6.2'},
  'sparql': {'version': 'sparql-1.1'},
  'opencypher': {'version': 'Neptune-9.0.20190305-1.0'},
  'labMode': {
    'ObjectIndex': 'disabled',
    'ReadWriteConflictDetection': 'enabled'
  },
  'features': {
    'SlowQueryLogs': 'disabled',
    'ResultCache': {'status': 'disabled'},
    'IAMAuthentication': 'disabled',
    'Streams': 'disabled',
    'AuditLog': 'disabled'
  },
  'settings': {
    'clusterQueryTimeoutInMs': '120000',
    'SlowQueryLogsThreshold': '5000'
```

```
  },
  'serverlessConfiguration': {
    'minCapacity': '1.0',
    'maxCapacity': '128.0'
  }
}
```

Se houver um problema com a instância, o comando de status retornará o código de erro `HTTP 500`. Se o host estiver inacessível, o tempo limite da solicitação será esgotado. Verifique se você está acessando a instância de dentro da nuvem privada virtual (VPC) e se seus grupos de segurança permitem o acesso a você.

Monitorando Neptune usando a Amazon CloudWatch

O Amazon Neptune e o CloudWatch Amazon são integrados para que você possa coletar e analisar métricas de desempenho. Você pode monitorar essas métricas usando o CloudWatch console, o AWS Command Line Interface (AWS CLI) ou a CloudWatch API.

CloudWatch também permite definir alarmes para que você possa ser notificado se um valor métrico ultrapassar um limite especificado por você. Você pode até mesmo configurar CloudWatch Eventos para tomar medidas corretivas caso ocorra uma violação. Para obter mais informações sobre uso CloudWatch e alarmes, consulte a CloudWatch documentação.

Tópicos

- Visualizando CloudWatch dados (console)
- Visualizando CloudWatch dados (AWS CLI)
- Visualização CloudWatch de dados (API)
- Usando CloudWatch para monitorar o desempenho da instância de banco de dados no Neptune
- Métricas de Neptune CloudWatch
- Dimensões de Netuno CloudWatch

Visualizando CloudWatch dados (console)

Para visualizar CloudWatch dados de um cluster do Neptune (console)

1. Faça login no AWS Management Console e abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.

2. No painel de navegação, selecione Métricas.
3. No painel Todas as métricas, escolha Neptune e, em seguida, escolha DB. ClusterIdentifier
4. No painel superior, role para baixo para visualizar a lista completa de métricas de seu cluster. As opções de métrica disponíveis do Neptune são exibidas na lista Visualização.

Para marcar ou desmarcar uma métrica individual, no painel de resultados, marque a caixa de seleção próxima ao nome do recurso e da métrica. Os gráficos que mostram as métricas para os itens selecionados aparecem na parte inferior do console. Para saber mais sobre CloudWatch gráficos, consulte Métricas gráficas no Guia do CloudWatch usuário da Amazon.

Visualizando CloudWatch dados (AWS CLI)

Para visualizar CloudWatch dados de um cluster de Neptune ()AWS CLI

1. Instale AWS CLI o. Para obter instruções, consulte o Guia do usuário AWS Command Line Interface.
2. Use o AWS CLI para buscar informações. Os CloudWatch parâmetros relevantes para Neptune estão listados em. Métricas de Neptune CloudWatch

   O exemplo a seguir recupera CloudWatch métricas para o número de solicitações do Gremlin por segundo para o cluster. `gremlin-cluster`

```
aws cloudwatch get-metric-statistics \
    --namespace AWS/Neptune  --metric-name GremlinRequestsPerSec \
    --dimensions Name=DBClusterIdentifier,Value=gremlin-cluster \
    --start-time 2018-03-03T00:00:00Z --end-time 2018-03-04T00:00:00Z \
    --period 60 --statistics=Average
```

Visualização CloudWatch de dados (API)

CloudWatch também oferece suporte a uma `Query` ação para que você possa solicitar informações programaticamente. Para obter mais informações, consulte a documentação da API CloudWatch Query e a Amazon CloudWatch API Reference.

Quando uma CloudWatch ação requer um parâmetro específico para o monitoramento de Netuno, `MetricName` como, use os valores listados em. Métricas de Neptune CloudWatch

O exemplo a seguir mostra uma CloudWatch solicitação de baixo nível, usando os seguintes parâmetros:

- `Statistics.member.1 = Average`
- `Dimensions.member.1 = DBClusterIdentifier=gremlin-cluster`
- `Namespace = AWS/Neptune`
- `StartTime = 2013-11-14T00:00:00Z`
- `EndTime = 2013-11-16T00:00:00Z`
- `Period = 60`
- `MetricName = GremlinRequestsPerSec`

Aqui está a aparência da CloudWatch solicitação. No entanto, a finalidade aqui é apenas mostrar o formulário da solicitação. Você deve criar a sua própria solicitação com base em suas métricas e períodos.

```
https://monitoring.amazonaws.com/
      ?SignatureVersion=2
      &Action=GremlinRequestsPerSec
      &Version=2010-08-01
      &StartTime=2018-03-03T00:00:00
      &EndTime=2018-03-04T00:00:00
      &Period=60
      &Statistics.member.1=Average
      &Dimensions.member.1=DBClusterIdentifier=gremlin-cluster
      &Namespace=AWS/Neptune
      &MetricName=GremlinRequests
      &Timestamp=2018-03-04T17%3A48%3A21.746Z
      &AWSAccessKeyId=AWS Access Key ID;
      &Signature=signature
```

Usando CloudWatch para monitorar o desempenho da instância de banco de dados no Neptune

Você pode usar CloudWatch métricas no Neptune para monitorar o que está acontecendo em suas instâncias de banco de dados e acompanhar o tamanho da fila de consultas conforme observado pelo banco de dados. As seguintes métricas são particularmente úteis:

- **`CPUUtilization`**: mostra o percentual de utilização da CPU.
- **`VolumeWriteIOPs`**: mostra o número médio de gravações de E/S do disco no volume de cluster, relatado em intervalos de cinco minutos.
- **`MainRequestQueuePendingRequests`**: mostra o número de solicitações na fila de entrada que aguardam execução.

Você também pode descobrir quantas solicitações estão pendentes no servidor usando o endpoint de status da consulta do Gremlin com o parâmetro `includeWaiting`. Isso fornecerá o status de todas as consultas em espera.

Os seguintes indicadores podem ajudar você a ajustar as estratégias de provisionamento e consulta do Neptune para melhorar a eficiência e o desempenho:

- Latência consistente, `CPUUtilization` alta, `VolumeWriteIOPs` altas e `MainRequestQueuePendingRequests` baixas juntas mostram que o servidor está ativamente envolvido no processamento de solicitações de gravação simultâneas a uma taxa sustentável, com pouca espera de E/S.
- Latência consistente, `CPUUtilization` baixa, `VolumeWriteIOPs` baixas e nenhuma `MainRequestQueuePendingRequests` juntas mostram que você tem excesso de capacidade na instância de banco de dados principal para processar solicitações de gravação.
- Latência de `CPUUtilization` e `VolumeWriteIOPs` altas, mas variáveis `MainRequestQueuePendingRequests` juntas, mostram que você está enviando mais trabalho do que o servidor pode processar em um intervalo específico. Pense em criar ou redimensionar solicitações em lote para realizar a mesma quantidade de trabalho com menos sobrecarga transacional e/ou aumentar a escala da instância principal verticalmente para aumentar o número de threads de consulta capazes de processar solicitações de gravação simultaneamente.
- A `CPUUtilization` baixa com as `VolumeWriteIOPs` altas significam que os threads de consulta estão aguardando a conclusão de operações de E/S na camada de armazenamento. Se você observar latências variáveis e algum aumento em `MainRequestQueuePendingRequests`, pense em criar ou redimensionar solicitações em lote para fazer a mesma quantidade de trabalho com menos sobrecarga transacional.

Métricas de Neptune CloudWatch

> **Note**
>
> O Amazon Neptune envia métricas somente quando elas têm CloudWatch um valor diferente de zero.
> Para todas as métricas do Neptune, a granularidade da agregação é de cinco minutos.

Tópicos

Métricas de Neptune CloudWatch

A tabela a seguir lista as CloudWatch métricas suportadas pelo Neptune.

> **Note**
>
> Todas as métricas cumulativas são zeradas sempre que o servidor é reiniciado, seja para manutenção, reinicialização ou recuperação de uma falha.

Métricas de Neptune CloudWatch

Métrica	Descrição
`BackupRetentionPeriodStorageUsed`	A quantidade total do armazenamento de backup, em bytes, usada para oferecer compatibilidade na janela de retenção do backup do cluster de banco de dados do Neptune. Incluído no total relatado pela métrica `TotalBackupStorageBilled` .
`BufferCacheHitRatio`	A porcentagem de solicitações atendidas pelo cache de buffer. Essa métrica pode ser útil para diagnosticar a latência da consulta, porque as falhas de cache induzem uma

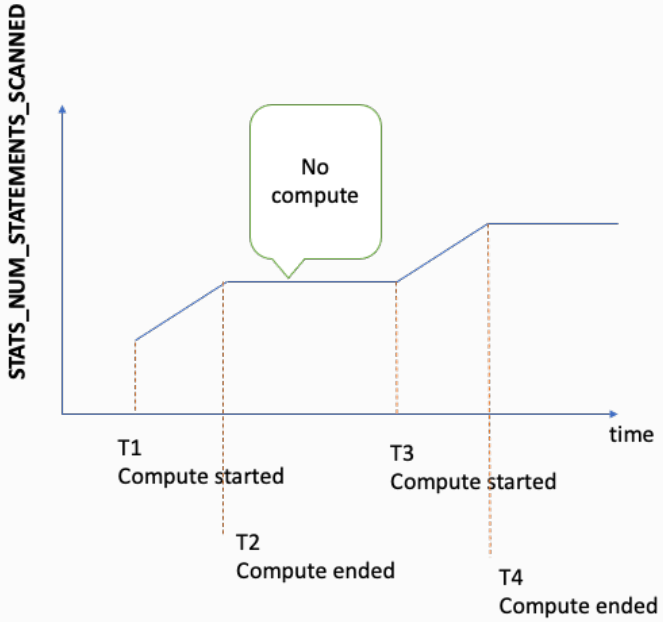
Métrica	Descrição
	latência significativa. Se a taxa de acertos do cache estiver abaixo de 99,9, pense em atualizar o tipo de instância para armazenar mais dados na memória.
`ClusterReplicaLag`	Para uma réplica de leitura, o tempo de atraso ao replicar atualizações da instância primária, em milissegundos.
`ClusterReplicaLagMaximum`	O tempo máximo de atraso entre a instância principal e cada instância de banco de dados do Neptune no cluster de banco de dados, em milissegundos.
`ClusterReplicaLagMinimum`	O tempo mínimo de atraso entre a instância principal e cada instância de banco de dados do Neptune no cluster de banco de dados, em milissegundos.
`CPUUtilization`	O percentual de utilização da CPU.
`EngineUptime`	A quantidade de tempo em que a instância está executando, em segundos.
`FreeableMemory`	A quantidade de memória de acesso aleatório disponível, em bytes.
`GlobalDbDataTransferBytes`	O número de bytes de dados de redo log transferidos do primário Região da AWS para o secundário Região da AWS em um banco de dados global do Neptune.

Métrica	Descrição
`GlobalDbReplicatedWriteIO`	O número de operações de E/S de gravação replicadas da região principal da Região da AWS para o volume do cluster em uma região secundária da Região da AWS. Os cálculos de cobrança para cada cluster de banco de dados em um banco de dados global do Neptune usam a métrica `VolumeWriteIOPS` para contabilizar as gravações realizadas nesse cluster. Para o cluster de banco de dados principal, os cálculos de cobrança usam `GlobalDbReplicatedWriteIO` para contabilizar a replicação entre regiões para clusters de banco de dados secundários.
`GlobalDbProgressLag`	O número de milissegundos em que um cluster secundário está atrás do cluster principal para transações de usuário e transações do sistema.
`GremlinRequestsPerSec`	Número de solicitações por segundo para o mecanismo Gremlin.
`GremlinWebSocketOpenConnections`	O número de WebSocket conexões abertas com Neptune.
`LoaderRequestsPerSec`	Número de solicitações do carregador por segundo.
`MainRequestQueuePendingRequests`	O número de solicitações na fila de entrada que aguardam execução. O Neptune começa a controlar a utilização das solicitações quando elas excedem a capacidade máxima da fila.

Métrica	Descrição
`NCUUtilization`	Aplicável somente a uma instância de banco de dados ou um cluster de banco de dados do Neptune Serverless. Em nível de instância, relata uma porcentagem calculada como o número de unidades de capacidade do Neptune (NCUs) usadas no momento pela instância em questão, dividida pela configuração de capacidade máxima de NCU do cluster. Uma NCU, ou unidade de capacidade do Neptune, consiste em 2 GiB (gibibyte) de memória (RAM), junto com a capacidade do processador virtual (vCPU) e a rede associadas. Em nível de cluster, `NCUUtilization` relata a porcentagem da capacidade máxima usada pelo cluster como um todo.
`NetworkThroughput`	A quantidade de throughput de rede recebida e transmitida aos clientes por cada instância no cluster de banco de dados do Neptune, em bytes por segundo. Esse throughput não inclui o tráfego de rede entre instâncias no cluster de banco de dados e o volume do cluster.
`NetworkTransmitThroughput`	A quantidade de throughput de rede recebida e transmitida aos clientes por cada instância no cluster de banco de dados do Neptune, em bytes por segundo. Esse throughput não inclui o tráfego de rede entre instâncias no cluster de banco de dados e o volume do cluster.
`NumTxCommitted`	O número de transações confirmadas com êxito por segundo.

Métrica	Descrição
`NumTxOpened`	O número de transações abertas no servidor por segundo.
`NumTxRolledBack`	Para consultas de gravação, o número de transações por segundo revertidas no servidor devido a erros. Para consultas somente leitura, essa métrica é igual ao número de transações somente leitura concluídas por segundo.
`OpenCypherRequestsPerSec`	Número de solicitações por segundo (HTTPS e Bolt) para o mecanismo do openCypher.
`OpenCypherBoltOpenConnections`	O número de conexões abertas do Bolt com o Neptune.
`ServerlessDatabaseCapacity`	Como métrica em nível de instância, `ServerlessDatabaseCapacity` relata a capacidade atual da instância de uma instância específica do Neptune Serverless, em NCUs. Uma NCU, ou unidade de capacidade do Neptune, consiste em 2 GiB (gibibyte) de memória (RAM), junto com a capacidade do processador virtual (vCPU) e a rede associada s. Em nível de cluster, o `ServerlessDatabaseCapacity` informa a média de todos os valores de `ServerlessDatabaseCapacity` das instâncias de banco de dados no cluster.

Métrica	Descrição
`SnapshotStorageUsed`	A quantidade total de armazenamento de backup consumida por todos os snapshots para um cluster de banco de dados do Neptune fora da janela de retenção de backup, em bytes. Incluído no total relatado pela métrica `TotalBackupStorageBilled` .
`SparqlRequestsPerSec`	Número de solicitações para o mecanismo do SPARQL por segundo.

Métrica	Descrição
`StatsNumStatementsScanned`	O número total de declarações verificadas para estatísticas do DFE desde a inicialização do servidor. Toda vez que o cálculo de estatísticas é acionado, esse número aumenta e, quando nenhum cálculo está acontecendo, ele permanece estático. Como resultado, se você representá-lo graficamente ao longo do tempo, poderá dizer quando a computação aconteceu e quando não aconteceu:  Ao observar a inclinação do grafo nos períodos em que a métrica está aumentando, você também pode ver a rapidez com que o cálculo estava ocorrendo. Se não houver essa métrica, isso significa que o atributo de estatísticas está desabilitado no cluster de banco de dados ou que a versão do mecanismo que você está executando não tem

Métrica	Descrição
	o atributo de estatísticas. Se o valor da métrica for zero, isso significa que não ocorreu nenhum cálculo de estatísticas.
`TotalBackupStorageBilled`	A quantidade total de armazenamento de backup pela qual você é cobrado para determinado cluster de banco de dados do Neptune, em bytes. Inclui o armazenam ento de backup medido pelas métricas `BackupRetentionPeriodStorageUsed` e `SnapshotStorageUsed` .
`TotalRequestsPerSec`	O número total de solicitações por segundo para o servidor de todas as origens.
`TotalClientErrorsPerSec`	O número total por segundo de solicitações que falharam devido a problemas no lado do cliente.
`TotalServerErrorsPerSec`	O número total por segundo de solicitações que falharam no servidor devido a falhas internas.

Métrica	Descrição
`UndoLogListSize`	A contagem de undo logs na lista de undo logs. Os undo logs contêm registros de transaçõe s confirmadas que expiram quando todas as transações ativas são mais recentes do que o tempo de confirmação. Os registros expirados são eliminados periodicamente. Os registros das operações de exclusão podem levar mais tempo para serem eliminados do que os registros de outros tipos de transação. A limpeza é realizada exclusivamente pela instância de gravador do cluster de banco de dados, portanto, a taxa de limpeza depende do tipo de instância de gravador. Se o valor `UndoLogListSize` estiver alto e crescendo no cluster de banco de dados, atualize a instância de gravador para aumentar a taxa de limpeza. Além disso, se você estiver realizando a atualização para a versão `1.2.0.0` do mecanismo ou posterior de uma versão anterior a `1.2.0.0`, primeiro verifique se o valor `UndoLogListSize` está próximo a 0. Como as versões do mecanismo `1.2.0.0` e posterior usam um formato diferente para undo logs, a atualização só pode começar depois que os undo logs anteriores forem totalment e eliminados. Consulte Realizar a atualização para a versão 1.2.0.0 ou posterior Para mais informações.

Métrica	Descrição
`VolumeBytesUsed`	A quantidade total de armazenamento alocada para o cluster de banco de dados do Neptune, em bytes. Este é o valor do armazenamento pelo qual você é cobrado. É a quantidade máxima de armazenamento alocada para seu cluster de banco de dados em qualquer ponto da sua existência, não a quantidade que você está usando atualmente (consulte Faturamento de armazenamento do Neptune).
`VolumeReadIOPs`	O número total de operações de E/S de leitura cobradas de um volume de cluster, relatado em intervalos de 5 minutos. As operações de leitura cobradas são calculadas em nível de volume do cluster, agregadas de todas as instâncias no cluster de banco de dados do Neptune e, depois, relatadas em intervalos de cinco minutos.
`VolumeWriteIOPs`	O número total de operações de E/S de disco de gravação no volume do cluster, relatado em intervalos de 5 minutos.

CloudWatch Métricas que agora estão obsoletas no Neptune

O uso dessas métricas do Neptune agora está obsoleto. Eles ainda têm suporte, mas poderão ser eliminados no futuro à medida que novas e melhores métricas se tornarem disponíveis.

Métrica	Descrição
`GremlinHttp1xx`	Número de respostas HTTP 1xx para o endpoint do Gremlin por segundo. Recomendamos o uso da nova métrica combinada `Http1xx` no seu lugar.

Métrica	Descrição
`GremlinHttp2xx`	Número de respostas HTTP 2xx para o endpoint do Gremlin por segundo. Recomendamos o uso da nova métrica combinada `Http2xx` no seu lugar.
`GremlinHttp4xx`	Número de erros HTTP 4xx para o endpoint do Gremlin por segundo. Recomendamos o uso da nova métrica combinada `Http4xx` no seu lugar.
`GremlinHttp5xx`	Número de erros HTTP 5xx para o endpoint do Gremlin por segundo. Recomendamos o uso da nova métrica combinada `Http5xx` no seu lugar.
`GremlinErrors`	Número de erros em percursos do Gremlin.
`GremlinRequests`	Número de solicitações para o mecanismo Gremlin.
`GremlinWebSocketSuccess`	Número de WebSocket conexões bem-suced idas com o endpoint Gremlin por segundo.
`GremlinWebSocketClientErrors`	Número de erros do WebSocket cliente no endpoint Gremlin por segundo.
`GremlinWebSocketServerErrors`	Número de erros WebSocket do servidor no endpoint do Gremlin por segundo.
`GremlinWebSocketAvailableCo nnections`	Número de WebSocket conexões potenciais atualmente disponíveis.

Métrica	Descrição
`Http100`	Número de respostas HTTP 100 para o endpoint por segundo. Recomendamos o uso da nova métrica combinada `Http1xx` no seu lugar.
`Http101`	Número de respostas HTTP 101 para o endpoint por segundo. Recomendamos o uso da nova métrica combinada `Http1xx` no seu lugar.
`Http1xx`	Número de respostas HTTP 1xx para o endpoint por segundo.
`Http200`	Número de respostas HTTP 200 para o endpoint por segundo. Recomendamos o uso da nova métrica combinada `Http2xx` no seu lugar.
`Http2xx`	Número de respostas HTTP 2xx para o endpoint por segundo.
`Http400`	Número de erros HTTP 400 para o endpoint por segundo. Recomendamos o uso da nova métrica combinada `Http4xx` no seu lugar.
`Http403`	Número de erros HTTP 403 para o endpoint por segundo. Recomendamos o uso da nova métrica combinada `Http4xx` no seu lugar.

Métrica	Descrição
`Http405`	Número de erros HTTP 405 para o endpoint por segundo. Recomendamos o uso da nova métrica combinada `Http4xx` no seu lugar.
`Http413`	Número de erros HTTP 413 para o endpoint por segundo. Recomendamos o uso da nova métrica combinada `Http4xx` no seu lugar.
`Http429`	Número de erros HTTP 429 para o endpoint por segundo. Recomendamos o uso da nova métrica combinada `Http4xx` no seu lugar.
`Http4xx`	Número de erros HTTP 4xx para o endpoint por segundo.
`Http500`	Número de erros HTTP 500 para o endpoint por segundo. Recomendamos o uso da nova métrica combinada `Http5xx` no seu lugar.
`Http501`	Número de erros HTTP 501 para o endpoint por segundo. Recomendamos o uso da nova métrica combinada `Http5xx` no seu lugar.
`Http5xx`	Número de erros HTTP 5xx para o endpoint por segundo.
`LoaderErrors`	Número de erros de solicitações do carregador.

Métrica	Descrição
`LoaderRequests`	Número de solicitações do carregador.
`SparqlHttp1xx`	Número de respostas HTTP 1xx para o endpoint do SPARQL por segundo. Recomendamos o uso da nova métrica combinada `Http1xx` no seu lugar.
`SparqlHttp2xx`	Número de respostas HTTP 2xx para o endpoint do SPARQL por segundo. Recomendamos o uso da nova métrica combinada `Http2xx` no seu lugar.
`SparqlHttp4xx`	Número de erros HTTP 4xx para o endpoint do SPARQL por segundo. Recomendamos o uso da nova métrica combinada `Http4xx` no seu lugar.
`SparqlHttp5xx`	Número de erros HTTP 5xx para o endpoint do SPARQL por segundo. Recomendamos o uso da nova métrica combinada `Http5xx` no seu lugar.
`SparqlErrors`	Número de erros nas consultas do SPARQL.
`SparqlRequests`	Número de solicitações para o mecanismo do SPARQL.
`StatusErrors`	Número de erros do endpoint de status.
`StatusRequests`	Número de solicitações para o endpoint de status.

Dimensões de Netuno CloudWatch

As métricas do Amazon Neptune são qualificadas de acordo com os valores da conta, do nome do grafo ou da operação. Você pode usar o CloudWatch console da Amazon para recuperar dados do Neptune junto com qualquer uma das dimensões na tabela a seguir.

Dimensão	Descrição
`DBInstanceIdentifier`	Filtra os dados solicitados por você para uma instância de banco de dados específica no cluster.
`DBClusterIdentifier`	Filtra os dados solicitados para um cluster de banco de dados específico do Neptune.
`DBClusterIdentifier` , `EngineName`	Filtra os dados de acordo com o cluster. O nome do mecanismo para todas as instâncias do Neptune é `neptune`.
`DBClusterIdentifier` , `Role`	Filtra os dados solicitados por você para um cluster de banco de dados específico do Neptune, agregando a métrica por perfil de instância (WRITER/READER). Por exemplo, é possível agregar métricas para todas as instâncias de READER que pertençam a um cluster.
`DBClusterIdentifier` , `SourceRegion`	Filtra os dados pelo cluster principal em uma região principal do banco de dados global.
`DatabaseClass`	Filtra os dados solicitados para todas as instâncias em uma classe de banco de dados. Por exemplo, você pode agregar métricas para todas as instâncias que pertencem à classe de banco de dados db.r4.large.
`EngineName`	O nome do mecanismo para todas as instância s do Neptune é `neptune`.

Dimensão	Descrição
`GlobalDbDBClusterIdentifier` , `SecondaryRegion`	Filtra os dados pelo cluster secundário de um banco de dados global especificado em uma região secundária.

Usar logs de auditoria com clusters do Amazon Neptune

Para auditar a atividade de cluster de banco de dados do Amazon Neptune, habilite a coleção de logs de auditoria configurando um parâmetro de cluster do banco de dados. Quando logs de auditoria são habilitados, você pode usá-los para registrar qualquer combinação de eventos compatíveis. Você pode visualizar ou fazer download dos logs de auditoria para analisá-los.

Habilitar os logs de auditoria do Neptune

Use o parâmetro `neptune_enable_audit_log` para habilitar (`1`) ou desabilitar (`0`) logs de auditoria.

Defina esse parâmetro no parameter group que é usado pelo cluster de banco de dados. Você pode usar o procedimento mostrado em Edição de um grupo de parâmetros de cluster de banco de dados ou de um grupo de parâmetros de banco de dados para modificar o parâmetro usando o AWS Management Console comando modify-db-cluster-parameter-group ou o AWS CLI comando da API ModifyDB Group para modificar o parâmetro programaticamente. ClusterParameter

É necessário reinicializar as instâncias de banco de dados depois de modificar esse parâmetro para que a alteração seja aplicada.

Visualizar logs de auditoria do Neptune usando o console

Você pode visualizar e fazer download dos logs de auditoria usando o AWS Management Console. Na página Instances (Instâncias), escolha a instância do banco de dados para exibir seus detalhes e, em seguida, vá até a seção Logs (Registros).

Para fazer download de um arquivo de log, selecione-o na seção Logs (Registros) e, depois, escolha Download (Fazer download).

Detalhes do log de auditoria do Neptune

Os arquivos de log estão em formato UTF-8. Os logs são gravados em vários arquivos, cujo número varia de acordo com o tamanho da instância. Para ver os eventos mais recentes, talvez seja preciso analisar todos os arquivos de log de auditoria.

As entradas do log não estão em ordem sequencial. Você pode usar o valor de `timestamp` para ordená-las.

Os arquivos de log são revezados quando atingem 100 MB em conjunto. Esse limite não é configurável.

Os arquivos de log de auditoria incluem as seguintes informações delimitadas por vírgulas em linhas, na ordem especificada:

Campo	Descrição
Timestamp	O time stamp do Unix para o evento registrado com precisão de microssegundo.
ClientHost	O nome de host ou IP no qual o usuário está conectado.
ServerHost	O nome de host ou IP da instância em que o evento foi registrado.
ConnectionType	O tipo de conexão. Pode ser `Websocket` , `HTTP_POST` , `HTTP_GET` ou `Bolt`.

ARN do IAM do chamador	O ARN do usuário do IAM ou do perfil do IAM usado para assinar a solicitação. Deverá ficar vazio se a autenticação do IAM estiver desabilitada. O formato é:
	`arn:`*`partition`*` :`*`service`*`:`*`region`*`:`*`account`*`:`*`resource`*
	Por exemplo: .
	`arn:aws:iam::123456789012:user/Anna`
	`arn:aws:sts::123456789012:assumed-role/AWSNeptuneNotebookRole/SageMaker`

Auth Context (Contexto de autenticação)	Contém um objeto JSON serializado com informações de autenticação. O campo `authenticationSucceeded` é `True` se o usuário fez a autenticação.

Campo	Descrição
	Deverá ficar vazio se a autenticação do IAM estiver desabilitada.
HttpHeader	As informações do cabeçalho HTTP. Pode conter uma consulta. Esvazie as conexões for WebSocket e Bolt.
Carga útil	A consulta do Gremlin, do SPARQL ou do openCypher.

Publicação do Neptune Logs no Amazon Logs CloudWatch

Você pode configurar um cluster de banco de dados Neptune para publicar dados de log de auditoria e/ou dados de log de consulta lenta em um grupo de logs no Amazon Logs. CloudWatch Com o CloudWatch Logs, você pode realizar análises em tempo real dos dados de registro e usá-los CloudWatch para criar alarmes e visualizar métricas. Você pode usar o CloudWatch Logs para armazenar seus registros de log em um armazenamento altamente durável.

Para publicar registros de auditoria no CloudWatch Logs, os registros de auditoria devem estar explicitamente ativados (consulteHabilitar logs de auditoria). Da mesma forma, para publicar registros de consulta lenta no Logs, CloudWatch os registros de consulta lenta devem estar explicitamente ativados (consulte). Usar o registro em log de consultas lentas do Amazon Neptune

> **Note**
>
> Esteja ciente do seguinte:
>
> - Cobranças adicionais se aplicam quando você publica registros no CloudWatch. Consulte a página CloudWatch de preços para obter detalhes.
> - Você não pode publicar CloudWatch registros em Registros da região da China (Pequim) ou China (Ningxia).
> - Se a exportação de dados de log estiver desabilitada, o Neptune não excluirá grupos de logs existentes nem fluxos de logs. Se a exportação de dados de registro estiver desativada, os dados de registro existentes permanecerão disponíveis em CloudWatch Registros, dependendo da retenção de registros, e você ainda incorrerá em cobranças pelos dados de registro de auditoria armazenados. Você pode excluir streams e grupos de registros usando o console de CloudWatch registros AWS CLI, o ou a API CloudWatch Logs.

Usando o console para publicar registros do Neptune em registros CloudWatch

Para publicar registros do Neptune no Logs CloudWatch a partir do console

1. Faça login no AWS Management Console e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. No painel de navegação, escolha Bancos de dados.
3. Selecione o cluster de banco de dados do Neptune no qual deseja publicar dados de log.
4. Para Actions (Ações), escolha Modify (Modificar).
5. Na seção Exportações de registros, escolha os registros que você quer começar a publicar no CloudWatch Logs.
6. Escolha Continue (Continuar) e depois escolha Modify DB Cluster (Modificar cluster de banco de dados) na página de resumo.

Usando a CLI para publicar registros de auditoria do Neptune no Logs CloudWatch

Você pode criar um novo cluster de banco de dados que publique registros de auditoria no CloudWatch Logs usando o AWS CLI `create-db-cluster` comando com os seguintes parâmetros:

```
aws neptune create-db-cluster \
    --region us-east-1  \
    --db-cluster-identifier my_db_cluster_id \
    --engine neptune \
    --enable-cloudwatch-logs-exports '["audit"]'
```

Você pode configurar um cluster de banco de dados existente para publicar registros de auditoria no CloudWatch Logs usando o AWS CLI `modify-db-cluster` comando com os seguintes parâmetros:

```
aws neptune modify-db-cluster \
    --region us-east-1  \
    --db-cluster-identifier my_db_cluster_id \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":["audit"]}'
```

Usando a CLI para publicar registros de consulta lenta do Neptune no Logs CloudWatch

Você também pode criar um novo cluster de banco de dados que publique registros de consulta lenta no Logs usando o AWS CLI `create-db-cluster` comando com os seguintes parâmetros: CloudWatch

```
aws neptune create-db-cluster \
    --region us-east-1  \
    --db-cluster-identifier my_db_cluster_id \
    --engine neptune \
    --enable-cloudwatch-logs-exports '["slowquery"]'
```

Da mesma forma, você pode configurar um cluster de banco de dados existente para publicar registros de consulta lenta no CloudWatch Logs usando o AWS CLI `modify-db-cluster` comando com os seguintes parâmetros:

```
aws neptune modify-db-cluster --region us-east-1  \
    --db-cluster-identifier my_db_cluster_id \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":["slowquery"]}'
```

Monitorando eventos do Neptune Log na Amazon CloudWatch

Depois de ativar os registros do Neptune, você pode monitorar eventos de log no Amazon Logs. CloudWatch Um novo grupo de logs é criado automaticamente para o cluster de banco de dados do Neptune com o seguinte prefixo, em que *`cluster-name`* representa o nome do cluster de banco de dados e *`log_type`* representa o tipo de log:

```
/aws/neptune/cluster-name/log_type
```

Por exemplo, se você configurar a função de exportação para incluir o log de auditoria de um cluster de banco de dados denominado `mydbcluster`, os dados de log serão armazenados no grupo de logs `/aws/neptune/mydbcluster/audit`.

Todos os eventos de todas as instâncias de banco de dados em um cluster de banco de dados são transferidos para um grupo de log usando diferentes fluxos de log.

Se já houver um grupo de logs com o nome especificado, o Neptune usará esse grupo para exportar dados de log para o cluster de banco de dados do Neptune. Você pode usar configurações

automatizadas, como, por exemplo AWS CloudFormation, para criar grupos de registros com períodos predefinidos de retenção de registros, filtros de métricas e acesso do cliente. Caso contrário, um novo grupo de registros será criado automaticamente usando o período padrão de retenção de registros, Never Expire, em CloudWatch Logs.

Você pode usar o console de CloudWatch registros AWS CLI, o ou a API de CloudWatch registros para alterar o período de retenção de registros. Para obter mais informações sobre como alterar os períodos de retenção de CloudWatch registros em registros, consulte Alterar retenção de dados de CloudWatch registros em registros.

Você pode usar o console CloudWatch Logs AWS CLI, o ou a API CloudWatch Logs para pesquisar informações nos eventos de log de um cluster de banco de dados. Para obter mais informações sobre como procurar e filtrar dados de log, consulte Procurar e filtrar dados de log.

Habilitando o Amazon CloudWatch Logs para um notebook Neptune

CloudWatch Os registros dos notebooks Neptune estão desativados por padrão. Siga estas etapas para habilitá-los, para depuração ou outros fins:

Usando o AWS Management Console para habilitar CloudWatch registros para um notebook Neptune

1. Abra o SageMaker console da Amazon em https://console.aws.amazon.com/sagemaker/.
2. No painel de navegação à esquerda, selecione Bloco de anotações e depois Instâncias do bloco de anotações. Procure o nome do bloco de anotações Neptune para o qual você gostaria de habilitar os logs.
3. Acesse a página de detalhes escolhendo o nome da instância do bloco de anotações mencionada na etapa acima.
4. Se a instância do bloco de anotações estiver em execução, selecione o botão Interromper, no canto superior direito da página de detalhes do bloco de anotações.
5. Em Permissões e criptografia, há um campo para o ARN do perfil do IAM. Selecione o link nesse campo para acessar o perfil do IAM desse bloco de anotações.
6. Crie a seguinte política:

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogDelivery",
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:DeleteLogDelivery",
        "logs:Describe*",
        "logs:GetLogDelivery",
        "logs:GetLogEvents",
        "logs:ListLogDeliveries",
        "logs:PutLogEvents",
        "logs:PutResourcePolicy",
        "logs:UpdateLogDelivery"
      ],
      "Resource": "*"
    }
  ]
}
```

7. Salve essa nova política e anexe-a ao perfil do IAM na etapa 4.
8. Selecione Iniciar no canto superior direito da página de detalhes da instância do SageMaker notebook.
9. Quando os logs começarem a fluir, você verá um link Visualizar logs abaixo do campo Configuração do ciclo de vida, na parte inferior esquerda da seção Configurações da instância do bloco de anotações na página de detalhes.

Se o notebook não iniciar, haverá uma mensagem na página de detalhes do notebook no SageMaker console informando que a instância do notebook demorou mais de 5 minutos para iniciar. Os CloudWatch registros relevantes para esse problema podem ser encontrados sob o nome:*(your-notebook-name)*`/LifecycleConfigOnStart`.

Consulte Registrar SageMaker eventos da Amazon com a Amazon CloudWatch para obter mais detalhes, se necessário.

Usar o registro em log de consultas lentas do Amazon Neptune

Identificar, depurar e otimizar uma consulta de execução lenta pode ser difícil. Quando o registro em log de consultas lentas do Neptune está habilitado, os atributos de todas as consultas de longa duração são registrados automaticamente para facilitar esse processo.

> **Note**
>
> O registro em log de consultas lentas foi introduzido na versão 1.2.1.0 do mecanismo do Neptune.

Você habilita o registro em log de consultas lentas usando o parâmetro de cluster de banco de dados neptune_enable_slow_query_log. Esse parâmetro é definido como `disabled` por padrão. Configurá-lo como `info` ou `debug` habilita o registro em log de consultas lentas. A configuração `info` registra alguns atributos úteis de cada consulta de execução lenta, enquanto a configuração `debug` registra todos os atributos disponíveis.

Para definir o limite para o que uma consulta seja considerada de execução lenta, use o parâmetro de cluster de banco de dados neptune_slow_query_log_threshold para especificar o número de milissegundos após os quais uma consulta em execução é considerada lenta e é registrada quando o registro em log de consultas lentas está habilitado. O valor padrão é cinco mil milissegundos (cinco segundos).

Você pode definir esses parâmetros do cluster de banco de dados no AWS Management Console, ou usando o comando AWS CLI modify-db-cluster-parameter-group, ou a função de gerenciamento do grupo ModifyDB. ClusterParameter

> **Note**
>
> Os parâmetros de registro em log de consultas lentas são dinâmicos, o que significa que a alteração dos valores não exige nem causa a reinicialização do cluster de banco de dados.

Para visualizar registros de consultas lentas no AWS Management Console

Você pode visualizar e baixar registros de consulta lenta no AWS Management Console, da seguinte forma:

Na página Instâncias, escolha a instância do banco de dados e, depois, vá até a seção Logs. Depois, você pode selecionar um arquivo de log e escolher Baixar para baixá-lo.

Os arquivos gerados pelo registro em log de consultas lentas do Neptune

Os arquivos de log gerados pelo registro em log de consultas lentas no Neptune têm as seguintes características:

- Os arquivos são codificados como UTF-8.
- As consultas e seus atributos são registrados no formato JSON.
- Atributos nulos e vazios não são registrados, exceto dados `queryTime`.
- Os logs abrangem vários arquivos, cujo número varia de acordo com o tamanho da instância.
- As entradas do log não estão em ordem sequencial. Você pode usar os valores `timestamp` para ordená-las.
- Para ver os eventos mais recentes, talvez seja preciso analisar todos os arquivos de log de consultas lentas.
- Os arquivos de log são trocados quando atingem 100 MiB em conjunto. Esse limite não é configurável.

Atributos de consulta registrados no modo **`info`**

Os seguintes atributos são registrados para consultas lentas quando o parâmetro do cluster de banco de dados `neptune_enable_slow_query_log` é definido como `info`:

Grupo	Atributo	Descrição
pedido ResponseMetadata	`requestId`	ID da solicitação da consulta.
	`requestType`	Tipo de solicitação, como HTTP ou WebSocket.
	`responseStatusCode`	Código de status da resposta da consulta, como 200.
	`exceptionClass`	Classe de exceção do erro gerado após a execução da consulta.

Grupo	Atributo	Descrição
queryStats	`query`	String de consulta.
	`queryFingerprint`	Impressão digital da consulta.
	`queryLanguage`	Linguagem de consulta, como Gremlin, SPARQL ou openCypher.
memoryStats	`allocatedPermits`	Permissões alocadas à consulta.
	`approximateUsedMemoryBytes`	Memória aproximada usada pela consulta durante a execução.
queryTime	`startTime`	Hora de início da consulta (UTC).
	`overallRunTimeMs`	Tempo total de execução da consulta, em milissegundos.
	`parsingTimeMs`	Tempo de análise da consulta, em milissegundos.
	`waitingTimeMs`	Tempo de espera na fila Gremlin/SPARQL/openCypher de consultas, em milissegundos
	`executionTimeMs`	Tempo de execução da consulta, em milissegundos.
	`serializationTimeMs`	Tempo de serialização da consulta, em milissegundos.
statementCounters	`scanned`	Número de declarações verificadas.

Grupo	Atributo	Descrição
	`written`	Número de declarações escritas.
	`deleted`	Número de declarações excluídas.
transactionCounters	`committed`	Número de transações confirmadas.
	`rolledBack`	Número de transações revertidas.
vertexCounters	`added`	Número de vértices adicionados.
	`removed`	Número de vértices removidos.
	`propertiesAdded`	Número de propriedades de vértice adicionadas.
	`propertiesRemoved`	Número de propriedades de vértice removidas.
edgeCounters	`added`	Número de bordas adicionadas.
	`removed`	Número de bordas removidas.
	`propertiesAdded`	Número de propriedades de borda adicionadas.
	`propertiesRemoved`	Número de propriedades de borda removidas.
resultCache	`hitCount`	Contagem de acertos do cache de resultados.

Grupo	Atributo	Descrição
	`missCount`	Contagem de erros do cache de resultados.
	`putCount`	Contagem de colocações do cache de resultados.
concurrentExecution	`acceptedQueryCountAtStart`	Consultas paralelas aceitas com a execução da consulta atual no início.
	`runningQueryCountAtStart`	Consultas paralelas em execução com a execução da consulta atual no início.
	`acceptedQueryCountAtEnd`	Consultas paralelas aceitas com a execução da consulta atual no fim.
	`runningQueryCountAtEnd`	Consultas paralelas em execução com a execução da consulta atual no fim.
queryBatch	`queryProcessingBatchSize`	Tamanho do lote durante o processamento da consulta.
	`querySerialisationBatchSize`	Tamanho do lote durante a serialização da consulta.

Atributos de consulta registrados no modo **debug**

Quando o parâmetro do cluster de banco de dados `neptune_enable_slow_query_log` é definido como `debug`, os seguintes atributos do contador de armazenamento são registrados, além dos atributos que são registrados como no modo `info`:

Atributo	Descrição
`statementsScannedInAllIndexes`	Declarações verificadas em todos os índices.
`statementsScannedSPOGIndex`	Declarações verificadas no índice SPOG.
`statementsScannedPOGSIndex`	Declarações verificadas no índice POGS.
`statementsScannedGPSOIndex`	Declarações verificadas no índice GPSO.
`statementsScannedOSGPIndex`	Declarações verificadas no índice OSGP.
`statementsScannedInChunk`	Declarações verificadas juntas no bloco.
`postFilteredStatementScans`	Declarações deixadas depois da pós-filtragem após as verificações.
`distinctStatementScans`	Declarações distintas verificadas.
`statementsReadInAllIndexes`	Declarações lidas depois da pós-filtragem em todos os índices.
`statementsReadSPOGIndex`	Declarações lidas depois da pós-filtragem no índice SPOG.
`statementsReadPOGSIndex`	Declarações lidas depois da pós-filtragem no índice POGS.
`statementsReadGPSOIndex`	Declarações lidas depois da pós-filtragem no índice GPSO.
`statementsReadOSGPIndex`	Declarações lidas depois da pós-filtragem no índice OSGP.
`accessPathSearches`	Número de pesquisas de caminhos de acesso.
`fullyBoundedAccessPathSearches`	Número de pesquisas de caminhos de acesso à chave totalmente limitada.

Atributo	Descrição
`accessPathSearchedByPrefix`	Número do caminho de acesso pesquisado por prefixo.
`searchesWhereRecordsWereFound`	Número de pesquisas que tiveram um ou mais registros como saída.
`searchesWhereRecordsWereNotFound`	Número de pesquisas que não tiveram registros como saída.
`totalRecordsFoundInSearches`	Total de registros encontrados em todas as pesquisas.
`statementsInsertedInAllIndexes`	Número de declarações inseridas em todos os índices.
`statementsUpdatedInAllIndexes`	Número de declarações atualizadas em todos os índices.
`statementsDeletedInAllIndexes`	Número de declarações excluídas em todos os índices.
`predicateCount`	Número de predicados.
`dictionaryReadsFromValueToIdTable`	Número de leituras do dicionário do valor até a tabela de ID.
`dictionaryReadsFromIdToValueTable`	Número de leituras do dicionário do ID da tabela de valores.
`dictionaryWritesToValueToIdTable`	Número de gravações do dicionário no valor na tabela de ID.
`dictionaryWritesToIdToValueTable`	Número de gravações do dicionário no ID na tabela de valores.
`rangeCountsInAllIndexes`	Número de contagens de intervalos em todos os índices.

Atributo	Descrição
`deadlockCount`	Número de deadlocks na consulta.
`singleCardinalityInserts`	Número de inserções de cardinalidade única realizadas.
`singleCardinalityInsertDeletions`	Número de declarações excluídas durante a inserção de cardinalidade única.

Exemplo de registro em log de depuração para uma consulta lenta

A seguinte consulta do Gremlin pode levar mais tempo para ser executada do que o limite definido para consultas lentas:

```
gremlin=g.V().has('code','AUS').repeat(out().simplePath()).until(has('code','AGR')).path().by('
```

Então, se o registro em log de consultas lentas estivesse habilitado no modo de depuração, os seguintes atributos seriam registrados para a consulta, da seguinte forma:

```
{
  "requestResponseMetadata": {
    "requestId": "5311e493-0e98-457e-9131-d250a2ce1e12",
    "requestType": "HTTP_GET",
    "responseStatusCode": 200
  },
  "queryStats": {
    "query":
 "gremlin=g.V().has('code','AUS').repeat(out().simplePath()).until(has('code','AGR')).path().by
    "queryFingerprint":
 "g.V().has(string0,string1).repeat(__.out().simplePath()).until(__.has(string0,string2)).path(
    "queryLanguage": "Gremlin"
  },
  "memoryStats": {
    "allocatedPermits": 20,
    "approximateUsedMemoryBytes": 14838
  },
  "queryTimeStats": {
    "startTime": "23/02/2023 11:42:52.657",
    "overallRunTimeMs": 2249,
```

```
    "executionTimeMs": 2229,
    "serializationTimeMs": 13
  },
  "statementCounters": {
    "read": 69979
  },
  "transactionCounters": {
    "committed": 1
  },
  "concurrentExecutionStats": {
    "acceptedQueryCountAtStart": 1
  },
  "queryBatchStats": {
    "queryProcessingBatchSize": 1000,
    "querySerialisationBatchSize": 1000
  },
  "storageCounters": {
    "statementsScannedInAllIndexes": 69979,
    "statementsScannedSPOGIndex": 44936,
    "statementsScannedPOGSIndex": 4,
    "statementsScannedGPSOIndex": 25039,
    "statementsReadInAllIndexes": 68566,
    "statementsReadSPOGIndex": 43544,
    "statementsReadPOGSIndex": 2,
    "statementsReadGPSOIndex": 25020,
    "accessPathSearches": 27,
    "fullyBoundedAccessPathSearches": 27,
    "dictionaryReadsFromValueToIdTable": 10,
    "dictionaryReadsFromIdToValueTable": 17,
    "rangeCountsInAllIndexes": 4
  }
}
```


Registro de chamadas de API do Amazon Neptune com AWS CloudTrail

O Amazon Neptune é integrado AWS CloudTrail com, um serviço que fornece um registro das ações realizadas por um usuário, função ou AWS serviço no Neptune. CloudTrail captura chamadas de API para o Neptune como eventos, incluindo chamadas do console do Neptune e de chamadas de código para as APIs do Neptune.

CloudTrail registra somente eventos para chamadas da API Neptune Management, como a criação de uma instância ou cluster. Se quiser auditar as alterações em seu gráfico, você poderá usar os logs de auditoria. Para ter mais informações, consulte Usar logs de auditoria com clusters do Amazon Neptune.

> **Important**
>
> As chamadas de console AWS CLI e API do Amazon Neptune são registradas como chamadas feitas para a API do Amazon Relational Database Service (Amazon RDS).

Se você criar uma trilha, poderá habilitar a entrega contínua de CloudTrail eventos para um bucket do Amazon S3, incluindo eventos para o Neptune. Se você não configurar uma trilha, ainda poderá ver os eventos mais recentes no CloudTrail console no Histórico de eventos. Usando as informações coletadas por CloudTrail, você pode determinar a solicitação que foi feita ao Neptune, o endereço IP do qual a solicitação foi feita, quem fez a solicitação, quando ela foi feita e detalhes adicionais.

Para saber mais sobre isso CloudTrail, consulte o Guia AWS CloudTrail do usuário.

Informações sobre Netuno em CloudTrail

CloudTrail é ativado em sua AWS conta quando você cria a conta. Quando a atividade ocorre no Amazon Neptune, essa atividade é registrada em CloudTrail um evento junto com AWS outros eventos de serviço no histórico de eventos. Você pode visualizar, pesquisar e baixar eventos recentes em sua AWS conta. Para obter mais informações, consulte Visualização de eventos com histórico de CloudTrail eventos.

Para um registro contínuo dos eventos em sua AWS conta, incluindo eventos de Neptune, crie uma trilha. Uma trilha permite CloudTrail entregar arquivos de log para um bucket do Amazon S3. Por padrão, ao criar uma trilha no console, a mesma é aplicada a todas as Regiões. A trilha registra eventos de todas as regiões na AWS partição e entrega os arquivos de log ao bucket do Amazon S3 que você especificar. Além disso, você pode configurar outros AWS serviços para analisar e agir com base nos dados de eventos coletados nos CloudTrail registros. Para obter mais informações, consulte:

- Visão geral da criação de uma trilha
- CloudTrail Serviços e integrações compatíveis
- Configurando notificações do Amazon SNS para CloudTrail

- Recebendo arquivos de CloudTrail log de várias regiões e recebendo arquivos de CloudTrail log de várias contas

Se uma ação for tomada em nome da sua AWS conta usando o console do Neptune, a interface da linha de comando do Neptune ou as APIs do SDK do Neptune, registrará a ação como chamadas feitas para a API AWS CloudTrail do Amazon RDS. Por exemplo, se você usa o console do Neptune para modificar uma instância de banco de dados ou chamar o comando modify-db-instance, AWS CLIoAWS CloudTrail log mostra uma chamada para a ação ModifyDBInstance da API do Amazon RDS. Para obter uma lista das ações da API Neptune registradas AWS CloudTrail, consulte a Referência da API Neptune.

> **Note**
>
> AWS CloudTrail registra somente eventos para chamadas da API Neptune Management, como a criação de uma instância ou cluster. Se quiser auditar as alterações em seu gráfico, você poderá usar os logs de auditoria. Para ter mais informações, consulte Usar logs de auditoria com clusters do Amazon Neptune.

Cada entrada de log ou evento contém informações sobre quem gerou a solicitação. As informações de identidade ajudam a determinar:

- Se a solicitação foi feita com credenciais de usuário raiz ou usuário do IAM.
- Se a solicitação foi feita com credenciais de segurança temporárias para um perfil ou usuário federado.
- Se a solicitação foi feita por outro AWS serviço.

Para obter mais informações, consulte o elemento CloudTrail UserIdentity.

Noções básicas das entradas dos arquivos de log do Neptune

Uma trilha é uma configuração que permite a entrega de eventos como arquivos de log para um bucket do Amazon S3 que você especificar. CloudTrail os arquivos de log contêm uma ou mais entradas de log. Um evento representa uma única solicitação de qualquer fonte e inclui informações sobre a ação solicitada, a data e a hora da ação, os parâmetros da solicitação e assim por diante. CloudTrail os arquivos de log não são um rastreamento de pilha ordenado das chamadas públicas de API, portanto, eles não aparecem em nenhuma ordem específica.

O exemplo a seguir mostra um CloudTrail registro de um usuário que criou um snapshot de uma instância de banco de dados e depois excluiu essa instância usando o console Neptune. O console é identificado pelo elemento `userAgent`. As chamadas de API solicitadas feitas pelo console (`CreateDBSnapshot` e `DeleteDBInstance`) são encontradas no elemento `eventName` para cada registro. Informações sobre o usuário (`Alice`) podem ser encontradas no elemento `userIdentity`.

```
{
  Records:[
  {
    "awsRegion":"us-west-2",
    "eventName":"CreateDBSnapshot",
    "eventSource":"",
    "eventTime":"2014-01-14T16:23:49Z",
    "eventVersion":"1.0",
    "sourceIPAddress":"192.0.2.01",
    "userAgent":"AWS Console, aws-sdk-java\/unknown-version Linux\/2.6.18-
kaos_fleet-1108-prod.2 Java_HotSpot(TM)_64-Bit_Server_VM\/24.45-b08",
    "userIdentity":
    {
      "accessKeyId":"",
      "accountId":"123456789012",
      "arn":"arn:aws:iam::123456789012:user/Alice",
      "principalId":"AIDAI2JXM4FBZZEXAMPLE",
      "sessionContext":
      {
        "attributes":
        {
          "creationDate":"2014-01-14T15:55:59Z",
          "mfaAuthenticated":false
        }
      },
      "type":"IAMUser",
      "userName":"Alice"
    }
  },
  {
    "awsRegion":"us-west-2",
    "eventName":"DeleteDBInstance",
    "eventSource":"",
    "eventTime":"2014-01-14T16:28:27Z",
    "eventVersion":"1.0",
    "sourceIPAddress":"192.0.2.01",
```

```
    "userAgent":"AWS Console, aws-sdk-java\/unknown-version Linux\/2.6.18-
kaos_fleet-1108-prod.2 Java_HotSpot(TM)_64-Bit_Server_VM\/24.45-b08",
    "userIdentity":
    {
      "accessKeyId":"",
      "accountId":"123456789012",
      "arn":"arn:aws:iam::123456789012:user/Alice",
      "principalId":"AIDAI2JXM4FBZZEXAMPLE",
      "sessionContext":
      {
        "attributes":
        {
          "creationDate":"2014-01-14T15:55:59Z",
          "mfaAuthenticated":false
        }
      },
      "type":"IAMUser",
      "userName":"Alice"
    }
  }
  ]
}
```

Usar a notificação de eventos do Neptune

Tópicos

- Categorias e mensagens de eventos do Amazon Neptune
- Assinar a notificação de eventos do Neptune
- Gerenciar assinaturas de notificação de eventos do Neptune

O Amazon Neptune usa o Amazon Simple Notification Service (Amazon SNS) para fornecer notificações quando ocorre um evento do Neptune. Essas notificações podem estar em qualquer formato compatível com o Amazon SNS para uma AWS região, como um e-mail, uma mensagem de texto ou uma chamada para um endpoint HTTP.

O Neptune agrupa esses eventos em categorias, que você pode assinar para receber notificações quando ocorrer um evento dessa categoria. É possível assinar uma categoria de evento para uma instância de banco de dados, um cluster de banco de dados, um snapshot de banco de dados, snapshot de cluster de banco de dados ou um grupo de parâmetros de banco de dados. Por

exemplo, se você se inscrever na categoria Backup para uma determinada instância de banco de dados, será notificado sempre que ocorrer um evento relacionado ao backup que afete a instância de banco de dados. Você também recebe uma notificação quando uma assinatura de notificação de evento é alterada.

Ocorrem eventos tanto no cluster de banco de dados quanto na instância de banco de dados, portanto, você poderá receber eventos se assinar um cluster ou uma instância de banco de dados.

Notificações de eventos são enviadas aos endereços fornecidos ao criar a assinatura. Pode ser interessante criar várias assinaturas diferentes, como uma assinatura que recebe todas as notificações de eventos e outra que inclua somente eventos críticos para as instâncias de bancos de dados de produção. Você pode facilmente desativar a notificação sem excluir uma assinatura. Para fazer isso, defina o botão de opção Habilitado como Não no console do Neptune.

> **Important**
>
> O Amazon Neptune não garante a ordem dos eventos enviados em um fluxo de eventos. A ordem do evento está sujeita a alterações.

O Neptune usa o nome do recurso da Amazon (ARN) de um tópico do Amazon SNS para identificar cada assinatura. O console do Neptune cria o ARN para você ao criar a assinatura.

O faturamento da notificação de eventos do Neptune é feito por meio do Amazon SNS. As taxas do Amazon SNS se aplicam durante o uso da notificação de eventos. Para obter mais informações, consulte Amazon Simple Notification Service Pricing.

Categorias e mensagens de eventos do Amazon Neptune

O Neptune gera um número significativo de eventos em categorias que você pode assinar usando o console do Neptune. Cada categoria se aplica a um tipo de origem, que pode ser uma instância de banco de dados, um snapshot de banco de dados ou um grupo de parâmetros de banco de dados.

> **Note**
>
> O Neptune usa IDs e definições de evento existentes do Amazon RDS.

Eventos do Neptune originados de instâncias de banco de dados

A tabela a seguir mostra uma lista de eventos por categoria de evento quando uma instância de banco de dados é o tipo de origem.

Categoria	ID do evento do Amazon RDS	Descrição
disponibilidade	RDS-EVENT-0006	A instância de banco de dados foi reiniciada.
	RDS-EVENT-0004	Desligamento da instância de banco de dados.
	RDS-EVENT-0022	Ocorreu um erro ao reiniciar o mecanismo do Neptune.
backup	RDS-EVENT-0001	Fazendo backup da instância de banco de dados.
	RDS-EVENT-0002	Backup da instância do banco de dados concluído.
alteração de configuração	RDS-EVENT-0009	A instância de banco de dados foi adicionada a um grupo de segurança.
	RDS-EVENT-0024	A instância de banco de dados está sendo convertida em uma instância de banco de dados multi-AZ.

Categoria	ID do evento do Amazon RDS	Descrição
	RDS-EVENT-0030	A instância do banco de dados está sendo convertida em uma instância de banco de dados single-AZ.
	RDS-EVENT-0012	Aplicando modificação à classe de instância de banco de dados.
	RDS-EVENT-0018	As configurações de armazenamento atuais para essa instância de banco de dados estão sendo alteradas.
	RDS-EVENT-0011	Um grupo de parâmetros para esta instância de banco de dados foi alterado.
	RDS-EVENT-0092	A atualização de um grupo de parâmetros para esta instância de banco de dados foi concluída.
	RDS-EVENT-0028	Os backups automáticos para esta instância de banco de dados foram desabilitados.

Categoria	ID do evento do Amazon RDS	Descrição
	RDS-EVENT-0032	Os backups automáticos para esta instância de banco de dados foram habilitados.
	RDS-EVENT-0025	A instância do banco de dados foi convertida em uma instância de banco de dados multi-AZ.
	RDS-EVENT-0029	A instância do banco de dados foi convertida em uma instância de banco de dados single-AZ.
	RDS-EVENT-0014	A classe de instância de banco de dados para essa instância de banco de dados foi alterada.
	RDS-EVENT-0017	As configurações de armazenamento para essa instância de banco de dados mudaram.
	RDS-EVENT-0010	A instância de banco de dados foi removida de um grupo de segurança.
criação	RDS-EVENT-0005	Instância de banco de dados criada.

Categoria	ID do evento do Amazon RDS	Descrição
exclusão	RDS-EVENT-0003	A instância de banco de dados foi excluída.
failover	RDS-EVENT-0034	O Neptune não está tentando um failover solicitado, pois ocorreu um failover na instância de banco de dados recentemente.
	RDS-EVENT-0013	Um failover de multi-AZ que resultou na promoção de uma instância em espera foi iniciado.
	RDS-EVENT-0015	Um failover de multi-AZ que resultou na promoção de uma instância em espera está concluído. Pode levar vários minutos para o DNS ser transferido para a nova instância primária do banco de dados.
	RDS-EVENT-0065	A instância se recuperou de um failover parcial.
	RDS-EVENT-0049	Um failover de multi-AZ foi concluído.

Categoria	ID do evento do Amazon RDS	Descrição
	RDS-EVENT-0050	Uma ativação multi-AZ foi iniciada após uma recuperação bem-sucedida da instância.
	RDS-EVENT-0051	Uma ativação multi-AZ está completa. Seu banco de dados deve estar acessível agora.
	RDS-EVENT-0031	Houve falha na instância de banco de dados devido a uma configuração incompatível ou a um problema do armazenamento subjacente. Comece a point-in-time-restore para a instância de banco de dados.
	RDS-EVENT-0036	A instância de banco de dados está em uma rede incompatí vel. Alguns dos IDs de sub-rede especific ados são inválidos ou não existem.

Categoria	ID do evento do Amazon RDS	Descrição
	RDS-EVENT-0035	A instância de banco de dados tem parâmetros inválidos . Por exemplo, se a instância de banco de dados não pôde ser iniciada porque um parâmetro relaciona do à memória está definido como um valor muito alto para essa classe de instância, a ação do cliente seria modificar o parâmetro da memória e reinicial izar a instância de banco de dados.
	RDS-EVENT-0082	O Neptune não pôde copiar dados de backup de um bucket do Amazon S3. É provável que as permissões para o Neptune acessar o bucket do Amazon S3 estejam configuradas incorretamente.

Categoria	ID do evento do Amazon RDS	Descrição
armazenamento baixo	RDS-EVENT-0089	A instância de banco de dados consumiu mais de 90% do armazenamento alocado. Você pode monitorar o espaço de armazenam ento para uma instância de banco de dados usando a métrica Free Storage Space (Espaço de armazenamento livre).
	RDS-EVENT-0007	O armazenamento alocado para a instância de banco de dados foi esgotado. Para resolver esse problema, você deve alocar armazenam ento adicional para a instância de banco de dados.
manutenção	RDS-EVENT-0026	Está ocorrendo a manutenção offline da instância de banco de dados. Atualmente, a instância do banco de dados está indisponí vel.

Categoria	ID do evento do Amazon RDS	Descrição
	RDS-EVENT-0027	A manutenção offline da instância de banco de dados está completa. A instância de banco de dados já está disponível.
	RDS-EVENT-0047	O patch da instância de banco de dados foi concluído.
notificação	RDS-EVENT-0044	Notificação emitida pelo operador. Para ter mais informações, consulte a mensagem do evento.
	RDS-EVENT-0048	O patch da instância de banco de dados foi atrasado.
	RDS-EVENT-0087	A instância do banco de dados foi interrompida.
	RDS-EVENT-0088	A instância do banco de dados foi iniciada.
	RDS-EVENT-0154	A instância de banco de dados está sendo iniciada porque excede o tempo máximo permitido para permanecer parada.

Categoria	ID do evento do Amazon RDS	Descrição
	RDS-EVENT-0158	A instância de banco de dados está em um estado que não pode ser atualizado.
	RDS-EVENT-0173	A instância de banco de dados foi corrigida.
réplica de leitura	RDS-EVENT-0045	Ocorreu um erro no processo de replicação de leitura. Para ter mais informações, consulte a mensagem do evento.

Categoria	ID do evento do Amazon RDS	Descrição
	RDS-EVENT-0046	A réplica de leitura retomou a replicação. Essa mensagem aparece quando você cria uma réplica de leitura pela primeira vez, ou como uma mensagem de monitoramento confirmando que a replicação está funcionando corretamente. Se essa mensagem seguir uma notificação RDS-EVENT-0045, a replicação foi retomada após um erro ou após a replicação ter sido interrompida.
	RDS-EVENT-0057	A replicação na réplica de leitura foi encerrada.
	RDS-EVENT-0062	A replicação na réplica de leitura foi interrompida manualmente.
	RDS-EVENT-0063	A replicação na réplica de leitura foi redefinida.

Categoria	ID do evento do Amazon RDS	Descrição
recuperação	RDS-EVENT-0020	A recuperação da instância de banco de dados começou. O tempo de recuperação variará dependendo da quantidade de dados a serem recuperados.
	RDS-EVENT-0021	A recuperação da instância de banco de dados está completa.
	RDS-EVENT-0023	Um backup manual foi solicitado, mas, no momento, o Neptune está no processo de criação um snapshot de banco de dados. Envie a solicitação novamente após o Neptune concluir o snapshot de banco de dados.
	RDS-EVENT-0052	A recuperação da instância multi-AZ começou. O tempo de recuperação variará dependendo da quantidade de dados a serem recuperados.

Categoria	ID do evento do Amazon RDS	Descrição
	RDS-EVENT-0053	A recuperação da instância multi-AZ está completa.
restauração	RDS-EVENT-0008	A instância de banco de dados foi restaurada a partir de um snapshot de banco de dados.
	RDS-EVENT-0019	A instância de banco de dados foi restaurada a partir de um point-in-time backup.

Eventos do Neptune originados de um cluster de banco de dados

A tabela a seguir mostra uma lista de eventos por categoria de evento quando um cluster de banco de dados é o tipo de origem.

Categoria	ID do evento do RDS	Descrição
failover	RDS-EVENT-0069	Um failover para o cluster de banco de dados falhou.
	RDS-EVENT-0070	Um failover para o cluster de banco de dados foi reiniciado.
	RDS-EVENT-0071	Um failover para o cluster de banco de dados foi concluído.
	RDS-EVENT-0072	Um failover para o cluster de banco

Categoria	ID do evento do RDS	Descrição
		de dados começou na mesma Zona de disponibilidade.
	RDS-EVENT-0073	Um failover para o cluster de banco de dados começou entre Zonas de disponibilidade.
	RDS-EVENT-0083	O Neptune não pôde copiar dados de backup de um bucket do Amazon S3. É provável que as permissões para o Neptune acessar o bucket do Amazon S3 estejam configuradas incorretamente.
manutenção	RDS-EVENT-0156	O cluster de banco de dados tem uma atualização de versão secundária do mecanismo de banco de dados disponível.
notificação	RDS-EVENT-0076	Falha na migração para um cluster de banco de dados do Neptune.

Categoria	ID do evento do RDS	Descrição
	RDS-EVENT-0077	Uma tentativa de converter uma tabela do banco de dados de origem para o formulário de banco de dados durante a migração para um cluster de banco de dados do Neptune.
	RDS-EVENT-0150	O cluster de banco de dados parou.
	RDS-EVENT-0151	O cluster de banco de dados iniciou.
	RDS-EVENT-0152	Falha na parada do cluster de banco de dados.
	RDS-EVENT-0153	O cluster de banco de dados está sendo iniciado porque excede o tempo máximo permitido para permanecer parado.

Eventos do Neptune originados de um snapshot de cluster de banco de dados

A tabela a seguir mostra a categoria de evento e uma lista de eventos quando um snapshot de cluster de banco de dados do Neptune é o tipo de origem.

Categoria	ID do evento do RDS	Descrição
backup	RDS-EVENT-0074	Foi iniciada a criação de um snapshot de cluster de banco de dados manual.
backup	RDS-EVENT-0075	Foi criado um snapshot de cluster de banco de dados manual.
notificação	RDS-EVENT-0162	Falha na tarefa de exportação do snapshot do cluster de banco de dados.
notificação	RDS-EVENT-0163	Tarefa de exportação de snapshot de cluster de banco de dados cancelada.
notificação	RDS-EVENT-0164	Tarefa de exportação de snapshot de cluster de banco de dados concluída.
backup	RDS-EVENT-0168	Criando snapshot de cluster automatizado.
backup	RDS-EVENT-0169	Snapshot de cluster automatizado criado.
criação	RDS-EVENT-0170	Cluster de banco de dados criado.
exclusão	RDS-EVENT-0171	Cluster de banco de dados excluído.

Categoria	ID do evento do RDS	Descrição
notificação	RDS-EVENT-0172	Cluster de banco de dados renomeado de [nome antigo do cluster de banco de dados] para [novo nome do cluster de banco de dados].

Eventos do Neptune originados de um grupo de parâmetros de banco de dados

A tabela a seguir mostra a categoria de evento e uma lista de eventos quando um grupo de parâmetros de cluster de banco de dados é o tipo de origem.

Categoria	ID do evento do RDS	Descrição
alteração de configuração	RDS-EVENT-0037	O grupo de parâmetros foi modificado.

Eventos do Neptune originados de um grupo de segurança

A tabela a seguir mostra a categoria de evento e uma lista de eventos quando um grupo de segurança é o tipo de origem.

Categoria	ID do evento do RDS	Descrição
alteração de configuração	RDS-EVENT-0038	O grupo de segurança foi modificado.
falha	RDS-EVENT-0039	O grupo de segurança de propriedade de [usuário] não existe; a autorização do grupo de segurança foi revogada.

Assinar a notificação de eventos do Neptune

É possível usar o console do Neptune para assinar notificações de eventos, da seguinte forma:

Para assinar a notificação de eventos do Neptune

1. Faça login no AWS Management Console e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. No painel de navegação, escolha Assinaturas de eventos.
3. No painel Event subscriptions (Assinaturas de eventos), escolha Create event subscription (Criar assinatura de evento).
4. Na caixa de diálogo Create event subscription (Criar assinatura de evento) faça o seguinte:
    a. Em Name (Nome), insira um nome para a assinatura de notificação de eventos.
    b. Em Send notifications to (Enviar notificações para), escolha um ARN do Amazon SNS existente para um tópico do Amazon SNS ou escolha create topic (criar tópico) para digitar o nome de um tópico e uma lista de destinatários.
    c. Em Source type (Tipo de origem), escolha um tipo de origem.
    d. Escolha Yes (Sim) para habilitar a assinatura. Se você quiser criar a assinatura, mas não tiver notificações enviadas, escolha No (Não).
    e. Dependendo do tipo de origem que você selecionou, escolha as categorias e as origens de eventos das quais quer receber notificações de eventos.
    f. Escolha Criar.

Gerenciar assinaturas de notificação de eventos do Neptune

Se você escolher Assinaturas de eventos no painel de navegação do console do Neptune, poderá visualizar as categorias de assinatura e uma lista das assinaturas atuais.

Você também pode modificar ou excluir uma assinatura específica.

Modificar assinaturas de notificação de eventos do Neptune

Como modificar assinaturas atuais de notificação de eventos do Neptune

1. Faça login no AWS Management Console e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.

2. No painel de navegação, escolha Event subscriptions (Assinaturas de eventos). O painel Event subscriptions (Assinaturas de eventos) exibirá todas as suas assinaturas de notificação de eventos.
3. No painel Event subscriptions (Assinaturas de eventos), escolha a assinatura que deseja modificar e escolha Edit (Editar).
4. Faça as alterações na assinatura usando as seções Target (Alvo) ou Source (Origem). É possível adicionar ou remover identificadores de origem selecionando-os ou desmarcando-os na seção Fonte.
5. Selecione a opção Editar. O console do Neptune indica que a assinatura está sendo modificada.

Excluir uma assinatura de notificação de eventos do Neptune

É possível excluir uma assinatura quando não precisar mais dela. Todos os assinantes do tópico não receberão mais notificações de evento especificadas pela assinatura.

Como excluir uma assinatura de notificação de eventos do Neptune

1. Faça login no AWS Management Console e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. No painel de navegação, escolha Assinaturas de eventos.
3. No painel Minhas assinaturas de eventos de banco de dados, escolha a assinatura que deseja excluir.
4. Escolha Excluir.
5. O console do Neptune indica que a assinatura está sendo excluída.

Marcar recursos do Amazon Neptune

Você pode usar as tags do Neptune para adicionar metadados aos recursos do Neptune. Além disso, você pode usar tags com políticas AWS Identity and Access Management (IAM) para gerenciar o acesso aos recursos do Neptune e controlar quais ações podem ser aplicadas a esses recursos. Finalmente, use essas tags para monitorar custos agrupando despesas de recursos marcados com tags semelhantes.

Todos os recursos administrativos do Neptune podem ser marcados, incluindo o seguinte:

- Instâncias de banco de dados

- clusters de banco de dados
- Réplicas de leitura
- DB snapshots
- Snapshots de cluster de banco de dados
- Assinaturas de eventos
- Grupos de parâmetros do banco de dados
- Grupos de parâmetros de cluster de banco de dados
- Grupos de sub-redes de banco de dados

Visão geral de tags dos recursos do Neptune

Uma tag do Amazon Neptune é um par de nome-valor definido e associado por você a um recurso do Neptune. O nome é referido como chave. Fornecer um valor para a chave é opcional. Você pode usar tags para atribuir informações arbitrárias a um recurso do Neptune. É possível usar uma chave de tag, por exemplo, para definir uma categoria, e o valor da tag pode ser um item nessa categoria. Por exemplo, você pode definir uma chave de tag como “projeto” e um valor de tag como “Salix”, indicando que o recurso do Neptune é atribuído ao projeto Salix. Também é possível usar tags para designar recursos do Neptune como sendo usados para testes ou produção usando uma chave como `environment=test` ou `environment=production`. Recomendamos que você use um conjunto consistente de chaves de tags para facilitar o monitoramento de metadados associados aos recursos do Neptune.

Use etiquetas para organizar sua AWS fatura de forma a refletir sua própria estrutura de custos. Para fazer isso, inscreva-se para receber sua Conta da AWS fatura com os valores-chave da etiqueta incluídos. Então, para ver o custo de recursos combinados, organize suas informações de faturamento de acordo com recursos com os mesmos valores de chave de tags. Por exemplo, é possível marcar vários recursos com um nome de aplicação específico, e depois organizar suas informações de faturamento para ver o custo total daquela aplicação em vários serviços. Para obter mais informações, consulte Usar tags de alocação de custos no Guia do usuário do AWS Billing .

Cada recurso do Neptune tem um conjunto de tags, que contém todas as tags atribuídas ao recurso do Neptune. Um conjunto de tags pode conter até 10 tags ou estar vazio. Se você adicionar uma tag a um recurso do Neptune que tenha a mesma chave que uma tag existente no recurso, o novo valor substituirá o antigo.

AWS não aplica nenhum significado semântico às suas tags; as tags são interpretadas estritamente como cadeias de caracteres. O Neptune pode definir tags em uma instância de banco de dados ou outros recursos do Neptune, dependendo das configurações usadas ao criar o recurso. Por exemplo, o Neptune pode adicionar uma tag indicando que uma instância de banco de dados é para produção ou para teste.

- A chave de tags é o nome obrigatório da tag. O valor da string pode ter de 1 a 128 caracteres Unicode e não pode ter os prefixos "`aws:`" ou "`rds:`". A string pode conter apenas o conjunto de letras Unicode, dígitos, espaço em branco,, '_', '.', '/', '=', '+', '-' (Java regex: "`^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$`").
- O valor da tag é um valor de string opcional da tag. O valor da string pode ter de 1 a 256 caracteres Unicode e não pode ter os prefixos "`aws:`". A string pode conter apenas o conjunto de letras Unicode, dígitos, espaço em branco,, '_', '.', '/', '=', '+', '-' (Java regex: "`^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$`").

  Os valores não têm que ser exclusivos em um conjunto de tags e podem ser nulos. Por exemplo, você pode ter um par de valor-chave em um conjunto de tag de `project/Trinity` e `cost-center/Trinity`.

> **Note**
>
> Você pode adicionar uma tag a um snapshot. No entanto, sua conta não refletirá esse agrupamento.

Você pode usar a AWS Management Console AWS CLI, a ou a API Neptune para adicionar, listar e excluir tags nos recursos do Neptune. Ao usar a AWS CLI ou a API Neptune, você deve fornecer o Amazon Resource Name (ARN) para o recurso Neptune com o qual você deseja trabalhar. Para obter mais informações sobre a criação de um ARN, consulte Criar um ARN para o Neptune.

As tags são armazenados em cache para finalidade de autorização. Por isso, as adições e as atualizações em tags dos recursos do Neptune podem demorar alguns minutos para ser disponibilizadas.

Copiar tags no Neptune

Ao criar ou restaurar uma instância de banco de dados, você pode especificar que as tags da instância de banco de dados sejam copiadas para snapshots da instância de banco de dados.

A cópia de tags garante que os metadados dos DB snapshots correspondam aos da instância de banco de dados de origem e que quaisquer políticas de acesso do DB snapshot também correspondam às da instância de banco de dados de origem. Tags não são copiadas por padrão.

Você pode especificar que as tags sejam copiados para snapshots de banco de dados para as seguintes ações:

- Criar uma instância de banco de dados.
- Restaurar uma instância de banco de dados.
- Criar uma réplica de leitura.
- Copiar um snapshot de banco de dados.

> **Note**
>
> Se você incluir um valor para o `--tag-key` parâmetro do AWS CLI comando create-db-cluster-snapshot (ou fornecer pelo menos uma tag para a CreateDBClusterSnapshot ação da API), o Neptune não copiará as tags da instância de banco de dados de origem para o novo DB snapshot. Isso será verdadeiro mesmo se a instância de banco de dados de origem tiver a opção `--copy-tags-to-snapshot` (`CopyTagsToSnapshot`) habilitada.
> Isso significa que você poderá criar uma cópia de uma instância de banco de dados com base em um snapshot de banco de dados sem adicionar tags que não se apliquem à nova instância de banco de dados. Depois de criar seu DB snapshot usando o AWS CLI `create-db-cluster-snapshot` comando (ou a `CreateDBClusterSnapshot` ação da API Neptune), você pode adicionar tags conforme descrito posteriormente neste tópico.

Marcando em Netuno usando o AWS Management Console

O processo para marcar um recurso do Amazon Neptune é semelhante para todos os recursos. O procedimento a seguir mostra como marcar uma instância de banco de dados do Neptune.

Para adicionar uma tag a uma instância de banco de dados

1. Faça login no AWS Management Console e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. No painel de navegação, escolha Instâncias.

> **Note**
>
> Para filtrar a lista de instâncias de bancos de dados no painel Instances (Instâncias), na caixa Filter instances (Filtrar instâncias), digite uma string de texto. Somente instâncias de banco de dados que contiverem a string aparecerão.

3. Escolha a instância de banco de dados que você deseja marcar.
4. Selecione Instance actions e, em seguida, See details.
5. Na seção de detalhes, role para baixo até a seção Tags.
6. Escolha Adicionar. A janela Add tags (Adicionar tags) é exibida.
7. Digite um valor para Tag key (Chave de tag) e para Value (Valor).
8. Para adicionar outra tag, você pode escolher Add another Tag (Adicionar outra tag) e digitar um valor para a Tag key (Chave de tag) e para Value (Valor).

   Repita esta etapa quantas vezes for necessário.
9. Escolha Adicionar.

Para excluir uma tag de uma instância de banco de dados

1. Faça login no AWS Management Console e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. No painel de navegação, escolha Instâncias.

> **Note**
>
> Para filtrar a lista de instâncias de bancos de dados no painel Instances (Instâncias), na caixa Filter instances (Filtrar instâncias), digite uma string de texto. Somente instâncias de banco de dados que contiverem a string aparecerão.

3. Escolha a instância de banco de dados que você deseja marcar.
4. Selecione Instance actions e, em seguida, See details.
5. Na seção de detalhes, role para baixo até a seção Tags.
6. Escolha a tag que você deseja excluir.
7. Escolha Remove (Remover) e depois Remove na janela Remove tags (Remover tags).

Marcando em Netuno usando o AWS CLI

É possível adicionar, listar ou remover tags de uma instância de banco de dados no Neptune usando a AWS CLI.

- Para adicionar uma ou mais tags a um recurso do Neptune, use o comando. AWS CLI `add-tags-to-resource`
- Para listar as tags em um recurso do Neptune, use o comando. AWS CLI `list-tags-for-resource`
- Para remover uma ou mais tags de um recurso do Neptune, use o comando. AWS CLI `remove-tags-from-resource`

Para saber mais sobre como criar o nome de recurso da Amazon (ARN) necessário, consulte Criar um ARN para o Neptune.

Marcar no Neptune usando a API

É possível adicionar, listar ou remover tags de uma instância de banco de dados usando a API do Neptune.

- Para adicionar uma tag a um recurso do Neptune, use a operação `AddTagsToResource`.
- Para listar tags atribuídas a um recurso do Neptune, use `ListTagsForResource`.
- Para remover tags de um recurso do Neptune, use a operação `RemoveTagsFromResource`.

Para saber mais sobre como criar o ARN necessário, consulte Criar um ARN para o Neptune.

As tags usam o seguinte esquema ao trabalhar com XML usando a API do Neptune:

```
<Tagging>
    <TagSet>
        <Tag>
            <Key>Project</Key>
            <Value>Trinity</Value>
        </Tag>
        <Tag>
            <Key>User</Key>
            <Value>Jones</Value>
        </Tag>
```

```
    </TagSet>
</Tagging>
```

A tabela a seguir fornece uma lista das tags XML permitidas e suas características. Os valores de `Key` e `Value` diferenciam letras maiúsculas e minúsculas. Por exemplo, `project=Trinity` e `PROJECT=Trinity` são duas tags distintas.

Elemento de marcação por tag	Descrição
`TagSet`	Conjunto de tags é um contêiner de todas as tags atribuídas a um recurso do Neptune. Só pode haver um conjunto de tags por recurso. Você trabalha com um `TagSet` somente por meio da API do Neptune.
`Tag`	Uma tag é um par de chave-valor definido pelo usuário. Pode haver de 1 a 50 tags em um conjunto de tags.
`Key`	Uma chave é o nome obrigatório da tag. O valor da string pode ter de 1 a 128 caracteres Unicode e não pode ter os prefixos "`rds:`" ou "`aws:`". A string pode conter apenas o conjunto de letras Unicode, dígitos, espaço em branco,, '_', '.', '/', '=', '+', '-' (Java regex: "`^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$`"). As chaves devem ser exclusivas a um conjunto de tags. Por exemplo, não pode haver um par de chaves em um conjunto de tags com a mesma chave com valores diferentes, como `project/Trinity` e `project/Xanadu`.
Valor	Um valor é o valor opcional da tag. O valor da string pode ter de 1 a 256 caracteres Unicode e não pode ter os prefixos "`rds:`" ou "`aws:`". A string pode conter apenas o conjunto de letras Unicode, dígitos, espaço em branco,, '_', '.', '/', '=', '+', '-' (Java regex: "`^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$`"). Os valores não têm de ser exclusivos em um conjunto de tags e podem ser nulos. Por exemplo, você pode ter um par de valor-chave em um conjunto de tag de `project/Trinity` e `cost-center/Trinity`.

Trabalhar com ARNs administrativos no Amazon Neptune

Os recursos criados na Amazon Web Services são identificados de forma exclusiva com um nome do recurso da Amazon (ARN). Para determinadas operações do Amazon Neptune, você deve identificar um recurso do Neptune de forma exclusiva especificando seu ARN.

> **Important**
>
> O Amazon Neptune compartilha o formato dos ARNs do Amazon RDS para ações administrativas que usam o Referência da API de gerenciamento. Os ARNs administrativos do Neptune contêm `rds` e não `neptune-db`. Para ARNs de plano de dados que identificam recursos de dados do Neptune, consulte Especificar recursos de dados.

Tópicos

Criar um ARN para o Neptune

Você pode criar um ARN para um recurso do Amazon Neptune usando a sintaxe a seguir. Observe que o Neptune compartilha o formato dos ARNs do Amazon RDS.

`arn:aws:rds:<region>:<account number>:<resourcetype>:<name>`

A tabela a seguir mostra o formato que deve ser usado ao criar um ARN para um tipo de recurso específico do Neptune.

Tipo de recurso	Formato do ARN
Instância de banco de dados	arn:aws:rds:*<region>*:*<account>* :db:*<name>* Por exemplo: . `arn:aws:rds: us-east-2 :123456789012 :db:my-instance-1`
Cluster de banco de dados	arn:aws:rds:*<region>*:*<account>* :cluster: *<name>*

Tipo de recurso	Formato do ARN
	Por exemplo: . `arn:aws:rds: us-east-2 :123456789012 :cluster: my-cluster-1`
Assinatura de eventos	arn:aws:rds:*<region>*:*<account>* :es:*<name>* Por exemplo: . `arn:aws:rds: us-east-2 :123456789012 :es:my-subscription`
DB parameter group (grupo de parâmetros de banco de dados)	arn:aws:rds:*<region>*:*<account>* :pg:*<name>* Por exemplo: . `arn:aws:rds: us-east-2 :123456789012 :pg:my-param-enable-logs`
Grupo de parâmetros do cluster de banco de dados	arn:aws:rds:*<region>*:*<account>* :cluster-pg: *<name>* Por exemplo: . `arn:aws:rds: us-east-2 :123456789012 :cluster-pg: my-cluster-param-timezone`
Snapshot de cluster de banco de dados	arn:aws:rds:*<region>*:*<account>* :cluster-snapshot: *<name>* Por exemplo: . `arn:aws:rds: us-east-2 :123456789012 :cluster-snapshot: my-snap-20160809`

Tipo de recurso	Formato do ARN
Grupo de sub-rede de banco de dados	arn:aws:rds:*<region>*:*<account>* :subgrp:*<name>* Por exemplo: . `arn:aws:rds: us-east-2 :123456789012 :subgrp:my-subnet -10`

Obter um ARN existente no Amazon Neptune

Você pode obter o ARN de um recurso do Neptune usando a API, AWS Management Console(AWS Command Line Interface)AWS CLI ou Neptune.

Obtendo um ARN existente usando o AWS Management Console

Para obter um ARN usando o console, navegue até o recurso para o qual deseja um ARN e veja os detalhes desse recurso. Por exemplo, para obter o ARN para uma instância de banco de dados, selecione Instances (Instâncias) no painel de navegação e selecione a instância que deseja na lista. O ARN está na seção Instance Details (Detalhes da instância).

Obtendo um ARN existente usando o AWS CLI

Para usar o AWS CLI para obter um ARN para um recurso específico do Neptune, use o comando para esse recurso. `describe` A tabela a seguir mostra cada AWS CLI comando e a propriedade ARN usados com o comando para obter um ARN.

AWS CLI Comando	Propriedade do ARN
describe-event-subscriptions	EventSubscriptionArn
describe-certificates	CertificateArn
describe-db-parameter-groups	Braço DB ParameterGroup
describe-db-cluster-parameter-groups	DB ClusterParameter GroupArn
describe-db-instances	DB InstanceArn

AWS CLI Comando	Propriedade do ARN
describe-events	SourceArn
describe-db-subnet-groups	Braço DB SubnetGroup
describe-db-clusters	DB ClusterArn
describe-db-cluster-snapshots	Braço DB ClusterSnapshot

Por exemplo, o AWS CLI comando a seguir obtém o ARN de uma instância de banco de dados.

Example

Para Linux, OS X ou Unix:

```
aws neptune describe-db-instances \
--db-instance-identifier DBInstanceIdentifier \
--region us-west-2
```

Para Windows:

```
aws neptune describe-db-instances ^
--db-instance-identifier DBInstanceIdentifier ^
--region us-west-2
```

Obter um ARN existente usando a API

Para obter um ARN para um recurso específico do Neptune, chame as seguintes ações da API e use as propriedades do ARN mostradas a seguir.

Ação da API Neptune	Propriedade do ARN
DescribeEventAssinaturas	EventSubscriptionArn
DescribeCertificates	CertificateArn
Descreva o DB ParameterGroups	Braço DB ParameterGroup

Ação da API Neptune	Propriedade do ARN
Grupos de banco de dados descritos ClusterParameter	DB ClusterParameter GroupArn
DescribeDBInstances	DB InstanceArn
DescribeEvents	SourceArn
Descreva o DB SubnetGroups	Braço DB SubnetGroup
DescribeDBClusters	DB ClusterArn
Descreva o DB ClusterSnapshots	Braço DB ClusterSnapshot

Fazer backup e restaurar um cluster de banco de dados do Amazon Neptune

Esta seção mostra como fazer backup e restaurar clusters de banco de dados do Amazon Neptune.

Tópicos

Visão geral do backup e da restauração de um cluster de banco de dados do Neptune

Esta seção fornece informações gerais sobre o backup e a restauração de dados no Amazon Neptune.

Tópicos

Tolerância a falhas para um cluster de banco de dados do Neptune

Um cluster de banco de dados do Neptune é tolerante a falhas por design. O volume do cluster abrange várias zonas de disponibilidade em uma única região da AWS e cada zona de disponibilidade contém uma cópia dos dados de volume do cluster. Esta funcionalidade significa que seu cluster de banco de dados pode tolerar falhas de uma Zona de disponibilidade sem perder dados, apenas uma breve interrupção do serviço.

Se a instância principal em um cluster de banco de dados falhar, o Neptune fará failover automaticamente para uma nova instância principal de duas maneiras:

- Ao promover uma réplica do Neptune existente para a nova instância principal.
- Ao criar uma nova instância primária

Se o cluster de banco de dados tiver uma ou mais réplicas do Neptune, uma réplica do Neptune será promovida à instância principal durante um evento de falha. Um evento de falha resulta em uma breve interrupção, durante a qual as operações de leitura e gravação falham com uma exceção. No entanto, o serviço é restaurado normalmente em menos de 120 segundos e muitas vezes em menos de 60 segundos. Para aumentar a disponibilidade do cluster de banco de dados, recomendamos que você crie pelo menos uma ou mais réplicas do Neptune em duas ou mais zonas de disponibilidade diferentes.

É possível personalizar a ordem em que as réplicas do Neptune são promovidas à instância principal após uma falha, atribuindo uma prioridade a cada réplica. As prioridades variam de 0 para a prioridade mais alta a 15 para a prioridade mais baixa. Se a instância principal falhar, o Neptune promoverá a réplica do Neptune com a prioridade mais alta à nova instância principal. É possível modificar a prioridade de uma réplica do Neptune a qualquer momento. Modificar a prioridade não desencadeia um failover.

É possível usar a AWS CLI para definir a prioridade de failover de uma instância de banco de dados da seguinte forma:

```
aws neptune modify-db-instance --db-instance-identifier (the instance ID) --promotion-
tier (the failover priority value)
```

A mesma prioridade pode ser compartilhada por mais de uma réplica do Neptune, gerando níveis de promoção. Se duas ou mais réplicas do Neptune compartilharem a mesma prioridade, o Neptune promoverá a réplica que for maior. Se duas ou mais réplicas do Neptune compartilharem a mesma prioridade e o mesmo tamanho, o Neptune promoverá uma réplica arbitrária no mesmo nível de promoção.

Se o cluster de banco de dados não contiver nenhuma réplica do Neptune, a instância principal será recriada durante um evento de falha. Um evento de falha resulta em uma interrupção durante a qual as operações de leitura e gravação falham com uma exceção. O serviço é reestabelecido quando a nova instância primária é criada, o que normalmente leva menos de 10 minutos. Promover uma réplica do Neptune à instância principal é muito mais rápido do que criar uma instância principal.

Backups do Neptune

O Neptune faz backup do volume de cluster automaticamente e mantém dados de restauração pela duração do período de retenção de backup. Os backups do Neptune são contínuos e incrementais para que você possa restaurar rapidamente em qualquer momento do período de retenção de backup. Quando os dados do backup estão sendo gravados, não há nenhum impacto sobre a performance ou interrupção de serviço do banco de dados. Você pode especificar um período de retenção de backup, de 1 a 35 dias, ao criar ou modificar um cluster de banco de dados.

Para controlar o uso de armazenamento de backup, reduza o intervalo de retenção de backup, remova snapshots manuais antigos quando eles não forem mais necessários, ou ambos. Para ajudar a gerenciar os custos, é possível monitorar a quantidade de armazenamento consumido por backups contínuos e snapshots manuais que são persistidos além do período de retenção. Você pode reduzir

o intervalo de retenção de backup e remover snapshots manuais quando eles não forem mais necessários.

Se você quiser manter um backup além do período de retenção do backup, também será possível fazer um snapshot dos dados no seu volume de cluster. Armazenar snapshots gera taxas de armazenamento padrão do Neptune. Para obter mais informações sobre preços de armazenamento do Neptune, consulte Preços do Amazon Neptune.

O Neptune retém dados de restauração incrementais durante todo o período de retenção de backup. Portanto, só será possível criar um snapshot para os dados que você deseja manter além desse período. Crie um novo cluster de banco de dados a partir do snapshot.

> **Important**
>
> Se você excluir um cluster de banco de dados, todos os seus backups automatizados serão excluídos ao mesmo tempo e não poderão ser recuperados. Isso significa que, a menos que você opte por criar um snapshot de banco de dados final manualmente, não poderá restaurar a instância de banco de dados para seu estado final posteriormente. Os snapshots manuais não são excluídos quando o cluster é excluído.

> **Note**
>
> - Para os clusters de banco de dados do Amazon Neptune, o período de retenção de backup padrão é de um dia, independentemente de como o cluster de banco de dados é criado.
> - Você não pode desabilitar backups automatizados no Neptune. O período de retenção de backup do Neptune é gerenciado pelo cluster de banco de dados.

Métricas do CloudWatch que são úteis para gerenciar o armazenamento de backup do Neptune

É possível usar as métricas `TotalBackupStorageBilled`, `SnapshotStorageUsed` e `BackupRetentionPeriodStorageUsed` do Amazon CloudWatch para examinar e monitorar a quantidade de armazenamento usado pelos backups do Neptune da seguinte forma:

- `BackupRetentionPeriodStorageUsed` representa a quantidade de armazenamento de backup usado, em bytes, para armazenar backups contínuos no momento atual. Esse valor depende do tamanho do volume do cluster e da quantidade de alterações feitas durante o período de retenção. No entanto, para fins de faturamento, ele não excede o tamanho cumulativo do volume do cluster durante o período de retenção. Por exemplo, se o tamanho de `VolumeBytesUsed` do cluster for 107.374.182.400 bytes (100 GiB), e o período de retenção for de dois dias, o valor máximo de `BackupRetentionPeriodStorageUsed` será 214.748.364.800 bytes (100 GiB + 100 GiB).
- `SnapshotStorageUsed` representa a quantidade de armazenamento de backup usado, em bytes, para armazenar snapshots manuais além do período de retenção de backup. Os snapshots manuais não contam em relação ao seu armazenamento de backups de snapshot enquanto o time stamp de criação estiver dentro do período de retenção. Todos os snapshots automáticos também não contam em relação ao armazenamento de backup. O tamanho de cada snapshot é o tamanho do volume do cluster no momento em que você faz o snapshot. O valor `SnapshotStorageUsed` depende do número de snapshots que você mantém e do tamanho de cada snapshot. Por exemplo, suponha que você tenha um snapshot fora do período de retenção, e o tamanho de `VolumeBytesUsed` do cluster era 100 GiB quando esse snapshot foi feito. A quantidade de SnapshotStorageUsed é 107.374.182.400 bytes (100 GiB).
- `TotalBackupStorageBilled` representa a soma, em bytes, de `BackupRetentionPeriodStorageUsed` e `SnapshotStorageUsed`, menos uma quantidade de armazenamento de backup grátis igual ao tamanho do volume do cluster para um dia. O armazenamento de backup gratuito é igual ao tamanho do volume mais recente. Por exemplo, se o tamanho de `VolumeBytesUsed` do cluster for 100 GiB, o período de retenção for de dois dias e você tiver um snapshot manual fora do período de retenção, o `TotalBackupStorageBilled` será de 214.748.364.800 bytes (200 GiB + 100 GiB - 100 GiB).

É possível monitorar os clusters do Neptune e criar relatórios usando métricas do CloudWatch por meio do console do CloudWatch. Para obter mais informações sobre como usar métricas do CloudWatch, consulte Monitorar o Neptune e a tabela de métricas em Métricas de Neptune CloudWatch .

Restaurar dados de um backup do Neptune

É possível recuperar os dados criando um cluster de banco de dados do Neptune por meio dos dados de backup retidos pelo Neptune ou por um snapshot de cluster de banco de dados que você salvou. É possível restaurar rapidamente uma nova cópia de um cluster de banco de dados criado

com os dados de backup a qualquer momento do período de retenção do backup. Devido à natureza contínua e incremental dos backups do Neptune durante o período de retenção do backup, você não precisa criar snapshots frequentes dos dados para melhorar os tempos de restauração.

Para determinar o tempo de restauração mais recente ou mais antigo de uma instância de banco de dados, procure os valores `Latest Restorable Time` ou `Earliest Restorable Time` no console do Neptune. O tempo de restauração mais recente de um cluster de banco de dados é o ponto mais recente no qual é possível restaurar seu cluster de banco de dados, normalmente dentro de 5 minutos do horário atual. O tempo de restauração mais antigo especifica o quão distante você pode restaurar o volume do cluster dentro do período de retenção do backup.

Você pode descobrir quando a restauração de um cluster de banco de dados foi encerrada verificando os valores `Latest Restorable Time` e `Earliest Restorable Time`. Os valores `Latest Restorable Time` e `Earliest Restorable Time` retornam NULL até que a operação de restauração seja concluída. Não é possível solicitar uma operação de backup ou restauração se `Latest Restorable Time` ou `Earliest Restorable Time` retornar NULL.

Para restaurar uma instância de banco de dados a um tempo especificado usando o AWS Management Console

1. Faça login no Console de Gerenciamento da AWS e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. No painel de navegação, escolha Instances (Instâncias). Selecione a instância primária do cluster de banco de dados que deseja restaurar.
3. Escolha Instance actions (Ações da instância) e, depois, Restore to point in time (Restauração point-in-time).

   Na janela Launch DB Instance (Iniciar instância de banco de dados), escolha Custom (Personalizado) em Restore time (Tempo de restauração).
4. Em Custom (Personalizado), especifique a data e a hora em que você quer realizar a restauração.
5. Digite o nome da sua nova instância de banco de dados restaurada em DB instance identifier (Identificador de instância de banco de dados) em Settings (Configurações).
6. Escolha Launch DB Instance (Iniciar instância de banco de dados) para executar a instância de banco de dados restaurada.

   São criados uma nova instância de banco de dados (com o nome que você especificou) e um novo cluster de banco de dados. O nome do cluster de banco de dados é o novo nome

da instância de banco de dados seguido por `-cluster`. Por exemplo, se o novo nome da instância de banco de dados for `myrestoreddb`, o nome do novo cluster de banco de dados será `myrestoreddb-cluster`.

Janela de backup no Neptune

Os backups automáticos são feitos diariamente durante a janela de backup escolhida. Se o backup exigir mais tempo do que o da janela de backup, ele continuará após a janela encerrar, até que esteja concluído. A janela de backup não pode se sobrepor à janela de manutenção semanal para a instância do banco de dados.

Durante a janela de backup automático, as E/S de armazenamento podem ser suspensas brevemente enquanto o processo de backup é inicializado (geralmente durante alguns segundos). Você pode perceber latências elevadas por alguns minutos durante os backups de implantações Multi-AZ.

A janela de backup é normalmente selecionada aleatoriamente em um bloco de tempo de oito horas por região pelo ambiente de gerenciamento do Amazon RDS subjacente ao Neptune. Os blocos de tempo de cada região da qual as janelas de backup padrão são atribuídas estão documentados na seção Backup Window do Guia do usuário do Amazon RDS.

Criar um snapshot de cluster de banco de dados no Neptune

O Neptune cria um snapshot do volume de armazenamento do cluster de banco de dados, fazendo backup de todo o cluster de banco de dados e não apenas dos bancos de dados individuais. Quando você cria um snapshot de cluster de banco de dados, precisa identificar de qual cluster de banco de dados fará backup. Em seguida, atribua um nome ao snapshot de cluster de banco de dados para que ele possa ser restaurado posteriormente. O tempo necessário para criar um snapshot do cluster de banco de dados varia de acordo com o tamanho dos bancos de dados. O snapshot inclui todo o volume de armazenamento. Portanto, o tamanho dos arquivos (como arquivos temporários) também afeta o tempo necessário para criar o snapshot.

É possível criar um snapshot do cluster de banco de dados usando o AWS Management Console, a AWS CLI ou a API do Neptune.

Usar o console para criar um snapshot de cluster de banco de dados

Para criar um snapshot de cluster de banco de dados

1. Faça login no Console de Gerenciamento da AWS e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. No painel de navegação, escolha Databases (Bancos de dados).
3. Na lista de instâncias de banco de dados, escolha a instância primária para o cluster de banco de dados.
4. Selecione Instance actions (Ações da instância) e, em seguida, selecione Take snapshot (Obter snapshot).

   A janela Take snapshot de banco de dados (Fazer snapshot de banco de dados) é exibida.
5. Insira o nome do snapshot do cluster de banco de dados na caixa Snapshot name (Nome do snapshot).
6. Escolha Take Snapshot (Obter Snapshot).

Restaurar a partir de um snapshot de cluster de banco de dados

Quando você cria um snapshot do Amazon Neptune de um cluster de banco de dados, o Neptune cria um snapshot do volume de armazenamento do cluster, fazendo backup de todos os dados e não apenas das instâncias individuais. É possível criar um cluster de banco de dados restaurando pelo snapshot desse cluster de banco de dados. Ao restaurar o cluster de banco de dados, você fornece o nome do snapshot do cluster de banco de dados do qual restaurar e um nome para o novo cluster de banco de dados criado pela restauração.

Sumário

Fatores a serem lembrados sobre a restauração de um cluster de banco de dados do Neptune por um snapshot

Não é possível realizar uma restauração para um cluster de banco de dados existente.

O processo de restauração sempre cria um cluster de banco de dados, então não é possível restaurar para um cluster de banco de dados que já exista.

Nenhuma instância é restaurada.

Um novo cluster de banco de dados criado por uma restauração não tem instâncias associadas a ele.

Assim que a restauração for concluída e o novo cluster de banco de dados estiver disponível, crie explicitamente as instâncias de que você precisará. Isso pode ser feito no console do Neptune ou usando a API CreateDBInstance.

Nenhum grupo de parâmetros personalizado é restaurado.

Um novo cluster de banco de dados criado por uma restauração tem automaticamente o grupo de parâmetros de banco de dados padrão associado a ele.

Assim que a restauração for concluída e o novo cluster do banco de dados estiver disponível, associe qualquer grupo de parâmetros de banco de dados personalizado usado pela instância da qual você restaurou. Para fazer isso, use o comando Modificar no console do Neptune ou na API ModifyDBInstance.

> **Important**
>
> Recomendamos salvar um grupo de parâmetros personalizado que esteja sendo usado em um cluster de banco de dados do qual você está criando um snapshot. Então, ao restaurar por esse snapshot, é possível associar facilmente o grupo de parâmetros correto ao cluster de banco de dados restaurado.

Nenhum grupo de segurança personalizado foi restaurado.

Um novo cluster de banco de dados criado por uma restauração tem automaticamente o grupo de segurança padrão associado a ele.

Assim que a restauração for concluída e o novo cluster de banco de dados estiver disponível, associe todos os grupos de segurança personalizados usados pela instância da qual a restauração foi feita. Para fazer isso, use o comando Modificar no console do Neptune ou na API ModifyDBInstance.

Não é possível restaurar por um snapshot compartilhado e criptografado.

Não é possível restaurar um cluster de banco de dados por um snapshot do cluster de banco de dados que seja compartilhado e criptografado.

Em vez disso, faça uma cópia não compartilhada do snapshot e restaure por ela.

Um cluster de banco de dados restaurado usa tanto armazenamento quanto antes.

Quando você restaura um cluster de banco de dados a partir de um snapshot de cluster de banco de dados, a quantidade de armazenamento alocada para o novo cluster é a mesma que foi alocada para o cluster de banco de dados do qual o snapshot foi criado, independentemente de quanto desse armazenamento alocado esteja realmente sendo usado.

Em outras palavras, a “marca d’água alta” pela qual você é cobrado não é alterada. A redefinição do nível máximo exige a exportação dos dados do gráfico e o recarregamento deles em um novo cluster de banco de dados (consulte Faturamento de armazenamento do Neptune).

Como restaurar por um snapshot

É possível restaurar um cluster de banco de dados por um snapshot de cluster de banco de dados usando o AWS Management Console, a AWS CLI ou a API do Neptune.

Usar o console para restaurar a partir de um snapshot

1. Faça login no Console de Gerenciamento da AWS e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. No painel de navegação, selecione Snapshots.
3. Escolha o snapshot do cluster de banco de dados a partir do qual você deseja restaurar.
4. Escolha Actions (Ações), Restore Snapshot (Restaurar snapshot).
5. Na página Restore DB Instance (Restaurar instância de banco de dados), na caixa DB Instance Identifier (Identificador da instância de banco de dados), digite o nome para o cluster de banco de dados restaurado.
6. Escolha Restore DB Instance.
7. Se quiser restaurar a funcionalidade do cluster de banco de dados para o cluster de banco de dados do qual o snapshot foi criado, você deverá modificar o cluster de banco de dados para usar o grupo de segurança. As próximas etapas supõem que o cluster de banco de dados esteja em uma nuvem privada virtual (VPC). Se o cluster de banco de dados não estiver em uma VPC, use o console do Amazon EC2 para localizar o grupo de segurança de banco de dados necessário para o cluster de banco de dados.

    a. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc/.

    b. No painel de navegação, selecione Grupos de segurança.

c. Selecione o grupo de segurança que você deseja usar para os clusters de banco de dados. Se necessário, adicione regras para vincular o security group a um security group de uma instância do EC2.

Cópia de um snapshot de cluster de banco de dados

Com o Neptune, é possível copiar snapshots manuais ou automatizados do cluster de banco de dados. Depois de copiar um snapshot, a cópia é um snapshot manual.

É possível copiar um snapshot na mesma região da AWS e entre regiões da AWS.

A cópia de um snapshot automatizado em outra conta da AWS é um processo de duas etapas. Primeiro, crie um snapshot manual pelo snapshot automatizado e, depois, copie o snapshot manual na outra conta.

Como alternativa à cópia, você também pode compartilhar snapshots manuais com outras contas da AWS. Para obter mais informações, consulte Compartilhar um snapshot do cluster de banco de dados.

Tópicos

Limitações para a cópia de um snapshot

Algumas limitações ao copiar snapshots:

- É possível copiar um snapshot entre China (Pequim) e China (Ningxia), mas não é possível copiar um snapshot entre essas regiões da China e outras da AWS.
- É possível copiar um snapshot entre a AWS GovCloud (Leste dos EUA) e a AWS GovCloud (Oeste dos EUA), mas não é possível copiar um snapshot entre essas regiões da AWS GovCloud (US) e outras da AWS.
- Se você excluir um snapshot de origem antes que o snapshot de destino fique disponível, a cópia do snapshot poderá falhar. Verifique se o snapshot de destino possui um status `AVAILABLE` antes de excluir um snapshot de origem.

- É possível ter até cinco solicitações de cópia de snapshot em andamento para uma única região por conta.
- Dependendo das regiões envolvidas e da quantidade de dados a serem copiados, uma cópia de snapshot entre regiões pode levar horas para ser concluída.

  Se houver um grande número de solicitações de cópia de snapshot entre regiões de uma região específica da AWS de origem, o Neptune pode colocar novas solicitações de cópia entre regiões dessa região da AWS de origem em uma fila até que algumas cópias em andamento sejam concluídas. Nenhuma informação de progresso sobre solicitações de cópia será exibida enquanto elas estiverem nessa fila. As informações de progresso são exibidas somente após o início da cópia.

Retenção das cópias de snapshot de cluster de banco de dados

O Neptune exclui snapshots automatizados da seguinte maneira:

- Ao final do período de retenção.
- Quando você desabilita snapshots automatizados para um cluster de banco de dados.
- Quando você exclui um cluster de banco de dados.

Se quiser manter um snapshot automatizado por um período mais longo, copie-o para criar um snapshot manual, que é mantido até você excluí-lo. Os custos de armazenamento do Neptune podem se aplicar a snapshots manuais, caso excedam o espaço de armazenamento padrão.

Para obter mais informações sobre os custos de armazenamento de backup, consulte Preços do Neptune.

Como lidar com a criptografia ao copiar snapshots

É possível copiar um snapshot que foi criptografado usando uma chave de criptografia do AWS KMS. Se você copiar um snapshot criptografado, a cópia desse snapshot também deverá ser criptografada. É possível criptografar a cópia com a mesma chave de criptografia do AWS KMS do snapshot original ou especificar outra chave de criptografia do AWS KMS.

Você não pode criptografar um snapshot de cluster de banco de dados não criptografado ao copiá-lo.

Para snapshots de cluster de banco de dados do Amazon Neptune, também é possível deixar o snapshot de cluster de banco de dados não criptografado e, em vez disso, especificar uma chave

de criptografia do AWS KMS ao restaurar. O cluster de banco de dados restaurado é criptografado usando a chave especificada.

Copiar snapshots entre regiões da AWS

> **Note**
>
> Esse atributo está disponível a partir da versão 1.0.2.1 do mecanismo do Neptune.

Quando você copia um snapshot em uma região da AWS que é diferente da região da AWS do snapshot de origem, a primeira cópia será uma cópia completa do snapshot, mesmo se você copiar um snapshot incremental. Uma cópia completa de snapshot contém todos os dados e metadados necessários para restaurar a instância de banco de dados. Após a primeira cópia do snapshot, você poderá copiar snapshots incrementais da mesma instância de bancos de dados na mesma região de destino dentro da mesma conta da AWS.

Um snapshot incremental contém somente os dados que foram alterados após o snapshot mais recente da mesma instância de banco de dados. A cópia incremental de snapshot é mais rápida e resulta em custos menores de armazenamento do que a cópia completa de snapshot. A cópia incremental de snapshot entre regiões da AWS é compatível com snapshots criptografados e não criptografados.

> **Important**
>
> Para snapshots compartilhados, a cópia incremental de snapshots não é compatível Para snapshots compartilhados, todas as cópias são snapshots completos, mesmo dentro da mesma região.

Dependendo das regiões da AWS envolvidas e da quantidade de dados a serem copiados, uma cópia de snapshot entre regiões pode levar horas para ser concluída.

Copiar um snapshot de cluster de banco de dados usando o console

Se o mecanismo de banco de dados de origem for o Neptune, o snapshot será de cluster de banco de dados. Para cada conta da AWS, é possível copiar até cinco snapshots de cluster de banco de dados por vez por região da AWS. A cópia de snapshots de cluster de banco de dados criptografados e não criptografados é aceita.

Para obter mais informações sobre preço da transferência de dados, consulte Preços do Neptune.

Para cancelar uma operação de cópia quando ela estiver em andamento, exclua o snapshot do cluster de banco de dados de destino enquanto ele estiver no status copying (cópia).

O procedimento a seguir funciona para copiar snapshots de cluster de banco de dados criptografados ou não criptografados:

Para copiar um snapshot de cluster de banco de dados

1. Faça login no Console de Gerenciamento da AWS e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. No painel de navegação, selecione Snapshots.
3. Marque a caixa de seleção para o snapshot de cluster de banco de dados que você deseja copiar.
4. Escolha Actions (Ações) e, em seguida, escolha Copy Snapshot (Copiar snapshot). A página Make Copy of snapshot de banco de dados (Copiar o snapshot de banco de dados) é exibida.
5. Digite o nome da cópia do snapshot de cluster de banco de dados em New DB Snapshot Identifier (Novo identificador de DB snapshot).
6. Para copiar tags e valores do snapshot para a cópia do snapshot, escolha Copy Tags (Copiar tags).
7. Em Enable Encryption (Habilitar criptografia), escolha uma das seguintes opções:
   - Escolha Disable encryption (Desabilitar criptografia) se o snapshot do cluster de banco de dados não estiver criptografado e você não quiser criptografar a cópia.
   - Escolha Enable encryption (Habilitar criptografia) se o snapshot do cluster de banco de dados não estiver criptografado, mas você quiser criptografar a cópia. Nesse caso, em Chave mestra, especifique o identificador de chave do AWS KMS a ser usado para criptografar a cópia do snapshot de cluster de banco de dados.
   - Escolha Enable encryption (Habilitar criptografia) se o snapshot do cluster de banco de dados estiver criptografado. Neste caso, você deve criptografar a cópia e, portanto, a opção Yes (Sim) já é selecionada. Em Chave mestra, especifique o identificador de chave do AWS KMS a ser usado para criptografar a cópia do snapshot de cluster de banco de dados.
8. Escolha Copy Snapshot (Copiar snapshot).

Copiar um snapshot de cluster de banco de dados usando a AWS CLI

É possível copiar um snapshot de banco de dados usando o comando copy-db-cluster-snapshot da AWS CLI.

Se você estiver copiando o snapshot em uma nova região da AWS, execute o comando na nova região.

Use as descrições e os exemplos de parâmetros a seguir para determinar quais parâmetros usar na cópia de um snapshot com a AWS CLI.

- `--source-db-cluster-snapshot-identifier` – O identificador do snapshot de banco de dados de origem.
  - Se o snapshot de origem estiver na mesma região da AWS que a cópia, especifique um identificador de snapshot de banco de dados válido, como `neptune:instance1-snapshot-20130805`.
  - Se o snapshot de origem estiver em uma região da AWS diferente da cópia, especifique um ARN de snapshot de banco de dados válido, como `arn:aws:neptune:us-west-2:123456789012:snapshot:instance1-snapshot-20130805`.
  - Se você estiver copiando a partir de um snapshot de banco de dados manual compartilhado, esse parâmetro deverá ser o nome de recurso da Amazon (ARN) desse snapshot.
  - Se você estiver copiando um snapshot criptografado, esse parâmetro deverá estar no formato ARN para a região da AWS de origem e deverá corresponder a `SourceDBSnapshotIdentifier` no parâmetro `PreSignedUrl`.
- `--target-db-cluster-snapshot-identifier`: o identificador da nova cópia do snapshot de banco de dados criptografado.
- `--kms-key-id`: o ID da chave do AWS KMS de um snapshot de banco de dados criptografado. O ID da chave do AWS KMS é o nome de recurso da Amazon (ARN), o identificador da chave do AWS KMS ou o alias da chave do AWS KMS para a chave de criptografia do AWS KMS.
  - Se você copiar um snapshot de banco de dados criptografado da conta da AWS, poderá especificar um valor para esse parâmetro a fim de criptografar a cópia com uma nova chave de criptografia do AWS KMS. Se você não especificar um valor para esse parâmetro, a cópia do snapshot de banco de dados será criptografada com a mesma chave do AWS KMS que o snapshot de banco de dados de origem.
  - Não é possível usar esse parâmetro para criar uma cópia criptografada de um snapshot não criptografado. Tentar fazer isso gerará um erro.

  - Ao copiar um snapshot criptografado em outra região da AWS, é necessário especificar uma chave do AWS KMS para a região da AWS de destino. Chaves de criptografia do AWS KMS são específicas da região da AWS em que são criadas, e não é possível usar chaves de criptografia de uma região da AWS em outra região da AWS.
- `--source-region`: o ID da região da AWS em que o snapshot de banco de dados de origem está. Se você copiar um snapshot criptografado para uma região da AWS diferente, deverá especificar essa opção.
- `--region`: o ID da região da AWS na qual você está copiando o snapshot. Se você copiar um snapshot criptografado para uma região da AWS diferente, deverá especificar essa opção.

Example De não criptografado, para a mesma região

O código a seguir cria uma cópia de um snapshot com o novo nome `mydbsnapshotcopy` da região `us-east-1` da AWS na região `us-west-2`.

Para Linux, OS X ou Unix:

```
aws neptune copy-db-cluster-snapshot \
  --source-db-cluster-snapshot-identifier instance1-snapshot-20130805 \
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy
```

Para Windows:

```
aws neptune copy-db-cluster-snapshot ^
  --source-db-cluster-snapshot-identifier instance1-snapshot-20130805 ^
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy
```

Example De não criptografado, entre regiões

O código a seguir cria uma cópia de um snapshot com o novo nome `mydbsnapshotcopy` da região `us-east-1` da AWS na região `us-west-2`. Execute o comando na região `us-west-2`.

Para Linux, OS X ou Unix:

```
aws neptune copy-db-cluster-snapshot \
  --source-db-cluster-snapshot-identifier arn:aws:neptune:us-
east-1:123456789012:snapshot:instance1-snapshot-20130805 \
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy \
  --source-region us-east-1 \
```

```
  --region us-west-2
```

Para Windows:

```
aws neptune copy-db-cluster-snapshot ^
  --source-db-cluster-snapshot-identifier arn:aws:neptune:us-
east-1:123456789012:snapshot:instance1-snapshot-20130805 ^
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy ^
  --source-region us-east-1 ^
  --region us-west-2
```

Example De criptografado, entre regiões

O exemplo de código a seguir copia um snapshot de banco de dados criptografado da região `us-east-1` da AWS na região `us-west-2`. Execute o comando na região `us-west-2`.

Para Linux, OS X ou Unix:

```
aws neptune copy-db-cluster-snapshot \
  --source-db-cluster-snapshot-identifier arn:aws:neptune:us-
west-2:123456789012:snapshot:instance1-snapshot-20161115 \
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy \
  --source-region us-east-1 \
  --region us-west-2
  --kms-key-id my_us_west_2_key
```

Para Windows:

```
aws neptune copy-db-cluster-snapshot ^
  --source-db-cluster-snapshot-identifier arn:aws:neptune:us-
west-2:123456789012:snapshot:instance1-snapshot-20161115 ^
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy ^
  --source-region us-east-1 ^
  --region us-west-2
  --kms-key-id my-us-west-2-key
```

Compartilhar um snapshot do cluster de banco de dados

Com o Neptune, é possível compartilhar um snapshot manual do cluster de banco de dados das seguintes formas:

- O compartilhamento de um snapshot manual de cluster de banco de dados, seja criptografado ou não, permite que as contas da AWS autorizadas copiem o snapshot.
- Compartilhar um snapshot de cluster de banco de dados manual, criptografado ou não, permite às contas da AWS autorizadas restaurarem diretamente um cluster de banco de dados a partir do snapshot em vez de fazer uma cópia dele e restaurar a partir daí.

> **Note**
>
> Para compartilhar um snapshot automatizado do cluster de banco de dados, crie um snapshot manual do cluster de banco de dados copiando o snapshot automatizado e, em seguida, compartilhe essa cópia.

Para mais informações sobre a restauração de um cluster de banco de dados a partir de um snapshot de cluster de banco de dados, consulte Como restaurar por um snapshot.

Você pode compartilhar um snapshot manual com até 20 outras contas da AWS. Também é possível compartilhar um snapshot manual não criptografado como público, disponibilizando-o para todas as contas da AWS. Ao fazer isso, tome cuidado para que suas informações privadas não estejam incluídas nos snapshots públicos.

> **Note**
>
> Ao restaurar um cluster de banco de dados por um snapshot compartilhado usando a AWS Command Line Interface (AWS CLI) ou a API do Neptune, é necessário especificar o nome do recurso da Amazon (ARN) do snapshot compartilhado como o identificador do snapshot.

Tópicos

Compartilhar um snapshot de cluster de banco de dados criptografado

Você pode compartilhar snapshots criptografados “em repouso” do cluster de banco de dados usando o algoritmo de criptografia AES-256. Para obter mais informações, consulte Criptografia de recursos em repouso do Neptune. Para fazer isso, você deve seguir as seguintes etapas:

1. Compartilhe a chave de criptografia do AWS Key Management Service (AWS KMS) que foi usada para criptografar o snapshot com todas as contas que você deseja que possam acessar o snapshot.

   É possível compartilhar chaves de criptografia do AWS KMS com outra conta da AWS adicionando a outra conta à política de chaves do KMS. Para obter mais detalhes sobre a atualização de uma política de chaves, consulte Políticas de chave no AWS KMSGuia do desenvolvedor do . Para ver um exemplo de como criar uma política de chaves, consulte Criação de uma política do IAM para permitir a cópia do snapshot criptografado, mais adiante neste tópico.
2. Use o AWS Management Console, a AWS CLI ou a API do Neptune para compartilhar o snapshot criptografado com outras contas.

Estas restrições se aplicam ao compartilhamento de snapshots criptografados:

- Não é possível compartilhar snapshots criptografados como públicos.
- Não é possível compartilhar um snapshot criptografado usando a chave de criptografia padrão do AWS KMS da conta da AWS que compartilhou o snapshot.

Permissão de acesso a uma chave de criptografia do AWS KMS

Para que outra conta da AWS possa copiar um snapshot do cluster de banco de dados criptografado compartilhado da conta, a conta com a qual você compartilhar o snapshot deverá ter acesso à chave do KMS que criptografou o snapshot. Para permitir o acesso de outra conta da AWS a uma chave do AWS KMS, atualize a política de chaves da chave do KMS, adicionando o ARN da conta da AWS com a qual você está compartilhando como `Principal` na política de chaves do KMS. Então, permita a ação `kms:CreateGrant`. Consulte Allowing users in other accounts to use a KMS key no Guia do desenvolvedor do AWS Key Management Service para obter instruções gerais.

Depois que você conceder a uma conta da AWS acesso à sua chave de criptografia do KMS, será necessário que essa conta da AWS crie um usuário do IAM se ainda não tiver um, para copiar o snapshot criptografado. As restrições de segurança do KMS não permitem o uso de uma identidade

de conta raiz da AWS para isso. A conta da AWS também deverá vincular uma política do IAM ao usuário do IAM para permitir que este copie um snapshot criptografado do cluster de banco de dados usando a chave do KMS.

No exemplo de política de chaves a seguir, o usuário `111122223333` é o proprietário da chave de criptografia do KMS, e o usuário `444455556666` é a conta com a qual a chave está sendo compartilhada. Essa atualização da política de chaves concede à conta da AWS acesso à chave do KMS ao incluir o ARN da identidade de conta raiz da AWS ao usuário `444455556666` como `Principal` na política e ao permitir a ação `kms:CreateGrant`.

```
{
  "Id=": "key-policy-1",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid=": "Allow use of the key",
      "Effect": "Allow",
      "Principal": {"AWS": [
        "arn:aws:iam::111122223333:user/KeyUser",
        "arn:aws:iam::444455556666:root"
      ]},
      "Action": [
        "kms:CreateGrant",
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    },
    {
      "Sid=": "Allow attachment of persistent resources",
      "Effect": "Allow",
      "Principal": {"AWS": [
        "arn:aws:iam::111122223333:user/KeyUser",
        "arn:aws:iam::444455556666:root"
      ]},
      "Action": [
        "kms:CreateGrant",
        "kms:ListGrants",
        "kms:RevokeGrant"
```

```
      ],
      "Resource": "*",
      "Condition": {"Bool": {"kms:GrantIsForAWSResource": true}}
    }
  ]
}
```

Criação de uma política do IAM para permitir a cópia do snapshot criptografado

Quando a conta da AWS externa já tiver acesso à chave do KMS, o proprietário dessa conta poderá criar uma política que permita a um usuário do IAM criado para essa conta copiar um snapshot criptografado com essa chave do KMS.

O exemplo a seguir mostra uma política que pode ser associada a um usuário do IAM para a conta da AWS `444455556666`. Ela permite que o usuário do IAM copie um snapshot compartilhado da conta da AWS `111122223333` que foi criptografada com a chave do KMS `c989c1dd-a3f2-4a5d-8d96-e793d082ab26` na região `us-west-2`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid=": "AllowUseOfTheKey",
            "Effect": "Allow",
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey",
                "kms:CreateGrant",
                "kms:RetireGrant"
            ],
            "Resource": ["arn:aws:kms:us-west-2:111122223333:key/c989c1dd-
a3f2-4a5d-8d96-e793d082ab26"]
        },
        {
            "Sid=": "AllowAttachmentOfPersistentResources",
            "Effect": "Allow",
            "Action": [
                "kms:CreateGrant",
                "kms:ListGrants",
                "kms:RevokeGrant"
```

```
            ],
            "Resource": ["arn:aws:kms:us-west-2:111122223333:key/c989c1dd-
a3f2-4a5d-8d96-e793d082ab26"],
            "Condition": {
                "Bool": {
                    "kms:GrantIsForAWSResource": true
                }
            }
        }
    ]
}
```

Para obter mais detalhes sobre a atualização de uma política de chaves, consulte Políticas de chave no AWS Key Management ServiceGuia do desenvolvedor do .

Compartilhamento de um snapshot do cluster de banco de dados

É possível compartilhar um snapshot do cluster de banco de dados usando o AWS Management Console, a AWS CLI ou a API do Neptune.

Usar o console para compartilhar um snapshot de cluster de banco de dados

Usando o console do Neptune, é possível compartilhar um snapshot manual do cluster de banco de dados com até vinte contas da AWS. Você também pode interromper o compartilhamento de um snapshot manual com uma ou mais contas.

Para compartilhar um snapshot de cluster de banco de dados manual

1. Faça login no Console de Gerenciamento da AWS e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. No painel de navegação, escolha Snapshots.
3. Escolha o snapshot manual que você deseja compartilhar.
4. Escolha Actions (Ações), Share Snapshot (Compartilhar snapshot).
5. Escolha uma das seguintes opções para DB snapshot visibility (Visibilidade do snapshot de banco de dados).

    - Se a origem não for criptografada, escolha Público para permitir que todas as contas da AWS restaurem um cluster de banco de dados pelo snapshot do cluster de banco de dados manual. Ou escolha Privado para permitir que apenas as contas da AWS que você especificar

restaurem um cluster de banco de dados por meio do snapshot de cluster de banco de dados manual.

> **Warning**
>
> Se você definir DB snapshot visibility (Visibilidade do snapshot do banco de dados) como Public (Pública), todas as contas da AWS poderão restaurar um cluster de banco de dados de um snapshot manual de cluster de banco de dados e ter acesso aos seus dados. Não compartilhe nenhum snapshot de cluster de banco de dados manual que contenha informações privadas, como Public (Público).

- Se a origem estiver criptografada, DB snapshot visibility (Visibilidade do snapshot de banco de dados) será definida como Private (Privada) porque os snapshots criptografados não podem ser compartilhados como públicos.

6. Em ID de conta da AWS, digite o identificador da conta da AWS à qual você deseja conceder a permissão para restaurar um cluster de banco de dados por meio do snapshot manual. Em seguida, escolha Adicionar. Repita a operação para incluir outros identificadores de contas da AWS, até chegar ao limite de 20 contas da AWS.

   Se você errar ao adicionar o identificador de conta da AWS à lista de contas permitidas, saiba que é possível excluí-lo da lista escolhendo Delete à direita do identificador incorreto da conta da AWS.
7. Depois de adicionar os identificadores de todas as contas da AWS às quais você deseja conceder a permissão para restaurar o snapshot manual, selecione Salvar.

Como interromper o compartilhamento de um snapshot manual de cluster de banco de dados com uma conta da AWS

1. Abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. No painel de navegação, escolha Snapshots.
3. Escolha o snapshot manual que você deseja interromper o compartilhamento.
4. Escolha Actions (Ações) e, em seguida, escolha Share Snapshot (Compartilhar snapshot).
5. Para remover a permissão de uma conta da AWS, escolha Delete para o identificador dessa conta da AWS na lista de contas autorizadas.
6. Escolha Save (Salvar).

Excluir um snapshot do Neptune

É possível excluir um snapshot de banco de dados usando o AWS Management Console, a AWS CLI ou a API de gerenciamento do Neptune:

Excluir usando o console

1. Faça login no Console de Gerenciamento da AWS e abra o console do Amazon Neptune em https://console.aws.amazon.com/neptune/home.
2. No painel de navegação, escolha Snapshots.
3. Escolha o snapshot de banco de dados que você deseja excluir.
4. Em Actions (Ações), escolha Delete Snapshot (Excluir snapshot).
5. Escolha Delete (Excluir) na página de confirmação.

Excluir usando a AWS CLI

Também é possível excluir um snapshot de banco de dados usando o comando delete_db_cluster_snapshot da AWS CLI usando o parâmetro `--db-snapshot-identifier` para identificar o snapshot que você deseja excluir:

Para Linux, OS X ou Unix:

```
aws neptune delete-db-cluster-snapshot \
    --db-snapshot-identifier <name-of-the-snapshot-to-delete>
```

Para Windows:

```
aws neptune delete-db-cluster-snapshot ^
    --db-snapshot-identifier <name-of-the-snapshot-to-delete>
```

Excluir usando a API de gerenciamento do Neptune

Você pode usar um dos SDKs para excluir um snapshot de banco de dados chamando a API DeleteDBClusterSnapshot e usar os parâmetros `DBSnapshotIdentifier` para identificar o snapshot de banco de dados a ser excluído.

Práticas recomendadas: aproveitar ao máximo o Neptune

Veja algumas recomendações gerais para trabalhar com o Amazon Neptune. Use essas informações como referência para localizar rapidamente as recomendações para o uso do Amazon Neptune e maximizar o desempenho.

Sumário

- Fechar o cliente para evitar o limite de conexões
- Criar uma conexão após o failover
- Definir maxInProcessPerConnection e maxSimultaneousUsagePerConnection como o mesmo valor
- Enviar consultas ao servidor como bytecode em vez de strings
- Sempre consuma completamente o ResultSet ou Iterator retornado por uma consulta
- Adicionar em massa vértices e bordas em lotes
- Desativar o armazenamento em cache DNS no Java Virtual Machine
- Opcionalmente, definir tempos limite em um nível por consulta
- Solução de problemas de java.util.concurrent.TimeoutException

- Práticas recomendadas para o Neptune ao usar openCypher e Bolt
  - Preferir direcionar para bordas bidirecionais nas consultas
  - O Neptune não é compatível com várias consultas simultâneas em uma transação.
  - Criar uma conexão após o failover
  - Tratamento de conexões para aplicações de longa duração
  - Manipulação de conexão para AWS Lambda
  - Fechar objetos de driver ao concluir
  - Usar modos de transação explícitos para leitura e gravação
    - Transações somente leitura
    - Transações somente leitura
  - Lógica de novas tentativas para exceções
  - Defina várias propriedades ao mesmo tempo usando uma única cláusula SET
  - Use a cláusula SET para remover várias propriedades de uma só vez
  - Use consultas parametrizadas
  - Use mapas achatados em vez de mapas aninhados na cláusula UNWIND
  - Coloque nós mais restritivos no lado esquerdo em expressões de caminho de comprimento variável (VLP)
  - Evite verificações redundantes de rótulos de nós usando nomes de relacionamento granulares
  - Especifique etiquetas de borda sempre que possível
  - Evite usar a cláusula WITH quando possível
  - Coloque filtros restritivos o mais cedo possível na consulta

Diretrizes operacionais básicas do Amazon Neptune

As diretrizes operacionais básicas a seguir devem ser seguidas ao trabalhar com o Neptune.

- Entenda as instâncias de banco de dados do Neptune para que você possa dimensioná-las adequadamente de acordo com seus requisitos de desempenho e caso de uso. Consulte Clusters e instâncias de banco de dados do Amazon Neptune.
- Monitore o uso que você faz da CPU e da memória. Isso ajuda você a saber quando migrar para uma classe de instância de banco de dados com maior capacidade de memória ou CPU para alcançar o desempenho necessário nas consultas. O Amazon CloudWatch pode ser configurado para lhe notificar quando os padrões de uso mudam ou quando você se aproxima da capacidade de implantação. Isso pode ajudá-lo a manter o desempenho e a disponibilidade do sistema. Consulte Monitorar instâncias e Monitorar o Neptune para obter mais detalhes.

  Como o Neptune tem seu próprio gerenciador de memória, é normal ver um uso relativamente baixo da memória, mesmo quando o uso da CPU é alto. A detecção de exceções de memória insuficiente ao executar consultas é o melhor indicador de que você precisa aumentar a memória passível de liberação.
- Ative os backups automáticos e defina a janela de backup para que ela ocorra em um momento conveniente.

- Teste o failover da instância do banco de dados para entender quanto tempo o processo leva para seu caso de uso. Isso também ajuda a garantir que o aplicativo que acessa sua instância de banco de dados possa se conectar automaticamente à nova instância de banco de dados após o failover.
- Se possível, execute o cliente e o cluster Neptune na mesma região e VPC, pois as conexões entre regiões com emparelhamento de VPC podem apresentar atrasos nos tempos de resposta de consulta. Para respostas de consulta em milissegundos de um único dígito, é necessário manter o cliente e o cluster do Neptune na mesma região e VPC.
- Quando você cria uma instância de réplica de leitura, ela deve ser pelo menos tão grande quanto a instância de gravador principal. Isso ajuda a manter o atraso de replicação sob controle e evita reinicializações de réplicas. Consulte Evitar classes de instância diferentes em um cluster.
- Antes de realizar a atualização para uma nova versão principal do mecanismo, teste a aplicação nela antes de fazer upgrade. Você pode fazer isso clonando o cluster de banco de dados para que o cluster clone execute a nova versão do mecanismo e, depois, testando a aplicação no clone.
- Para facilitar os failovers, é ideal que todas as instâncias tenham o mesmo tamanho.

Tópicos

Práticas recomendadas de segurança para o Amazon Neptune

Use contas do AWS Identity and Access Management (IAM) para controlar o acesso a ações de API do Neptune. Controle ações que criam, modificam ou excluem recursos do Neptune (como instâncias

de banco de dados, grupos de segurança, grupos de opções ou grupos de parâmetros) e ações administrativas comuns (como fazer backup e restaurar instâncias de banco de dados).

- Use credenciais temporárias em vez de persistentes sempre que possível.
- Atribua uma conta do IAM individual a cada pessoa que gerencia recursos do Amazon Relational Database Service (Amazon RDS). Nunca utilize usuários raiz da conta da AWS para gerenciar os recursos do Neptune. Crie um usuário do IAM para todos os usuários, incluindo você mesmo.
- Conceda a cada usuário o conjunto mínimo de permissões necessárias para realizar suas funções.
- Use grupos do IAM para gerenciar efetivamente permissões para vários usuários.
- Mude suas credenciais do IAM regularmente.

Para obter mais informações sobre o uso do IAM para acessar os recursos do Neptune, consulte Segurança no Amazon Neptune. Para obter informações gerais sobre como trabalhar com o IAM, consulte AWS Identity and Access Management e IAM Best Practices no Guia do usuário do IAM.

Evitar classes de instância diferentes em um cluster

Quando o cluster de banco de dados contém instâncias de classes diferentes, podem ocorrer problemas com o tempo. O problema mais comum é que uma pequena instância de leitor pode entrar em um ciclo de reinicializações repetidas devido ao atraso na replicação. Se um nó de leitor tiver uma configuração de classe de instância de banco de dados mais fraca do que a de uma instância de banco de dados de gravador, o volume de alterações poderá ser muito grande para o leitor se atualizar.

> **Important**
>
> Para evitar reinicializações repetidas causadas pelo atraso na replicação, configure o cluster de banco de dados para que todas as instâncias tenham a mesma classe (tamanho) de instância.

É possível ver o atraso entre a instância de gravador (a principal) e os leitores no cluster de banco de dados usando a métrica `ClusterReplicaLag` no Amazon CloudWatch. A métrica `VolumeWriteIOPs` também permite detectar picos de atividade de gravação no cluster que podem criar atrasos na replicação.

Evitar reinicializações repetidas durante o carregamento em massa

Se você tiver um ciclo de reinicializações repetidas de réplica de leitura em virtude do atraso na replicação durante o carregamento em massa, é provável que as réplicas não consigam acompanhar o gravador no cluster de banco de dados.

Escale os leitores para serem maiores do que o gravador ou remova-os temporariamente durante o carregamento em massa e, depois, recrie-os após a conclusão.

Habilitar o Índice OSGP se você tiver um grande número de predicados

Se o seu modelo de dados contiver um grande número de predicados distintos (mais de mil na maioria dos casos), o desempenho poderá ser degradado e os custos operacionais mais altos.

Se for esse o caso, você poderá melhorar o desempenho habilitando o índice OSGP. Consulte O índice OSGP.

Evitar transações de longa execução quando possível

Transações de longa execução, somente leitura ou leitura e gravação, podem causar problemas inesperados dos seguintes tipos:

Uma transação de longa execução em uma instância de leitor ou em uma instância de gravador com gravações simultâneas pode ocasionar um grande acúmulo de diferentes versões de dados. Isso pode introduzir latências mais altas para consultas de leitura que filtrem uma grande parte dos resultados.

Em alguns casos, as versões acumuladas ao longo de horas podem fazer com que novas gravações sejam limitadas.

Uma transação de leitura e gravação de longa execução com muitas gravações também poderá causar problemas se a instância for reiniciada. Se uma instância for reiniciada a partir de um evento de manutenção ou de uma falha, todas as gravações não confirmadas serão revertidas. Essas operações de desfazer normalmente são executadas em segundo plano e não impedem que a instância volte a funcionar, mas qualquer nova gravação que entre em conflito com as operações que estão sendo revertidas falhará.

Por exemplo, se a mesma consulta for repetida após a conexão ter sido interrompida na execução anterior, ela poderá falhar quando a instância for reiniciada.

O tempo necessário para as operações de desfazer é proporcional ao tamanho das alterações envolvidas.

Práticas recomendadas para o uso de métricas do Neptune

Para identificar problemas de desempenho causados por recursos insuficientes e outros gargalos comuns, é possível monitorar as métricas disponíveis para o cluster de banco de dados do Neptune.

Monitore as métricas de desempenho regularmente para coletar dados sobre os valores médio, máximo e mínimo de uma série de intervalos de tempo. Isso ajuda a identificar quando o desempenho está degradado. Usando esses dados, você também pode definir alarmes do Amazon CloudWatch para limites métricos específicos para que você seja alertado se eles forem atingidos.

Quando você configura um novo cluster de banco de dados e a executa com uma carga de trabalho típica, tente captar os valores médio, máximo e mínimo de todas as métricas de desempenho em vários intervalos diferentes (por exemplo, uma hora, 24 horas, uma semana, duas semanas). Isso dá a você uma ideia do que é normal. Isso ajuda a obter comparações para as horas de operação de pico e fora de pico. Você pode usar essas informações para identificar quando o desempenho está ficando abaixo dos níveis padrão e definir alarmes corretamente.

Consulte Monitorando Neptune usando a Amazon CloudWatch para obter informações sobre como visualizar métricas do Neptune.

Veja a seguir as métricas mais importantes para começar:

- BufferCacheHitRatio: a porcentagem de solicitações atendidas pelo cache de buffer. As falhas de cache adicionam latência significativa à execução da consulta. Se a taxa de acertos do cache estiver abaixo de 99,9% e a latência for um problema na aplicação, pense em atualizar o tipo de instância para armazenar em cache mais dados na memória.
- Utilização da CPU: porcentagem da capacidade de processamento computacional utilizada. Altos valores de consumo de CPU podem ser adequados, dependendo de seus objetivos de desempenho de consultas.
- Memória disponível: quanta RAM está disponível na instância de banco de dados, em megabytes. O Neptune tem seu próprio gerenciador de memória, então essa métrica pode ser mais baixa do que você espera. Um bom sinal de que você deve considerar atualizar sua instância para uma classe com mais memória RAM é se as consultas geralmente lançam exceções de memória insuficiente.

A linha vermelha nas métricas da guia Monitoring (Monitoramento) é marcada em 75% para CPU e métricas de memória. Se o consumo de memória da instância cruzar essa linha com frequência, verifique sua carga de trabalho ou considere atualizar sua instância para melhorar o desempenho das consultas.

Práticas recomendadas para ajustar as consultas do Neptune

Uma das melhores maneiras de melhorar o desempenho do Neptune é ajustar as consultas mais utilizadas e que requerem mais recursos para baixar o custo de execução delas.

Para obter informações sobre como ajustar as consultas do Gremlin, consulte Dicas de consulta do Gremlin e Ajustar consultas do Gremlin. Para obter informações sobre como ajustar consultas do SPARQL, consulte Dicas de consulta do SPARQL.

Balanceamento de carga entre réplicas de leitura

O roteamento ida e volta do endpoint de leitor funciona alterando o host para o qual a entrada DNS aponta. O cliente deve criar uma nova conexão e resolver o registro de DNS para obter uma conexão com uma nova réplica de leitura, já que as conexões WebSocket são frequentemente mantidas ativas por longos períodos.

Para obter réplicas de leitura diferentes para solicitações sucessivas, certifique-se de que o cliente resolva a entrada de DNS sempre que se conectar. Isso pode exigir o fechamento da conexão e a reconexão ao endpoint de leitor.

Você também pode balancear a carga de solicitações entre réplicas de leitura conectando-se aos endpoints da instância explicitamente.

Carregar com maior rapidez usando uma instância temporária maior

O desempenho do carregamento aumenta com tamanhos de instância maiores. Se não estiver usando um tipo de instância ampla, mas quer aumentar a velocidade de carregamento, você pode usar a instância ampla para carregar e então excluí-la.

> **Note**
>
> O procedimento a seguir é para um novo cluster. Se tiver um cluster existente, você pode adicionar uma nova instância maior e, em seguida, promovê-la para uma instância de Banco de Dados.

Para carregar dados usando um tamanho de instância maior

1. Crie um cluster com uma única instância `r5.12xlarge`. Esta instância é a principal instância de banco de dados.
2. Crie uma ou mais réplicas de leitura do mesmo tamanho (`r5.12xlarge`).

   É possível criar as réplicas de leitura em um tamanho menor, mas se elas não forem grandes o suficiente para acompanhar as gravações feitas pela instância principal, talvez precisem ser reiniciadas com frequência. O tempo de inatividade resultante reduz drasticamente o desempenho.
3. No comando do carregador em massa, inclua `“parallelism” : “OVERSUBSCRIBE”` para indicar ao Neptune que use todos os recursos de CPU disponíveis para carregamento (consulte Parâmetros de solicitação do carregador do Neptune). A operação de carregamento prosseguirá tão rápido quanto a E/S permitir, o que geralmente requer de 60% a 70% dos recursos da CPU.
4. Carregue os dados usando o carregador do Neptune. O trabalho de carga é executado na instância de banco de dados principal.
5. Depois que os dados terminarem de carregar, certifique-se de reduzir todas as instâncias no cluster para o mesmo tipo de instância a fim de evitar cobranças adicionais e problemas repetidos de reinicialização (consulte Evitar tamanhos de instância diferentes.).

Redimensione a instância de gravador realizando o failover para uma réplica de leitura

A melhor maneira de redimensionar uma instância no cluster de banco de dados, incluindo a instância de gravador, é criar ou modificar uma instância de réplica de leitura para que ela tenha o tamanho desejado e, depois, fazer o failover deliberadamente para essa réplica de leitura. O tempo de inatividade observado pela aplicação é apenas o tempo necessário para alterar o endereço IP do gravador, que deve ser de cerca de três a cinco segundos.

A API de gerenciamento do Neptune usada para fazer failover deliberadamente da instância de gravador atual para uma instância de réplica de leitura é FailoverDBCluster. Se você estiver usando o cliente Java do Gremlin, talvez seja necessário criar um objeto Client após o failover para obter o novo endereço IP, conforme mencionado aqui.

Altere todas as instâncias para o mesmo tamanho a fim de evitar um ciclo de reinicializações repetidas, conforme mencionado abaixo.

Tentar fazer upload novamente após um erro de interrupção de tarefa de pré-busca de dados

Ocasionalmente, quando você estiver carregando dados no Neptune usando o carregador em massa, poderá ocorrer um status `LOAD_FAILED`, com uma mensagem `PARSING_ERROR` e `Data prefetch task interrupted` relatadas em resposta a uma solicitação de informações detalhadas, como:

```
"errorLogs" : [
  {
    "errorCode" : "PARSING_ERROR",
    "errorMessage" : "Data prefetch task interrupted: Data prefetch task for 11467
 failed",
    "fileName" : "s3://some-source-bucket/some-source-file",
    "recordNum" : 0
  }
]
```

Se você encontrar esse erro, basta executar novamente a solicitação de upload em massa.

O erro ocorre quando há uma interrupção temporária que normalmente não foi causada por sua solicitação ou seus dados e, geralmente, ele pode ser resolvido executando a solicitação de upload em massa novamente.

Se você estiver usando as configurações padrão, ou seja, `"mode":"AUTO"` e `"failOnError":"TRUE"`, o carregador ignorará os arquivos que ele já tiver carregado com êxito e retomará o carregamento de arquivos que ainda não tinha carregado quando a interrupção ocorreu.

Práticas recomendadas gerais para usar o Gremlin com o Neptune

Siga estas recomendações ao usar a linguagem de percurso de grafos Gremlin com o Neptune. Para obter informações sobre como usar o Gremlin com Neptune, consulte the section called “Gremlin”.

> **Important**
>
> Foi feita uma alteração na versão 3.4.11 do TinkerPop que melhora a exatidão de como as consultas são processadas, mas, no momento, às vezes pode afetar gravemente o desempenho das consultas.
>
> Por exemplo, uma consulta desse tipo pode apresentar uma lentidão significativa:

```
g.V().hasLabel('airport').
  order().
    by(out().count(),desc).
  limit(10).
  out()
```

Os vértices após a etapa limite agora são buscados de uma forma não ideal devido à alteração do TinkerPop 3.4.11. Para evitar isso, é possível modificar a consulta adicionando a etapa barrier() a qualquer momento após `order().by()`. Por exemplo:

```
g.V().hasLabel('airport').
  order().
    by(out().count(),desc).
  limit(10).
  barrier().
  out()
```

O TinkerPop 3.4.11 foi habilitado na versão 1.0.5.0 do mecanismo do Neptune.

Tópicos

- Testar o código do Gremlin no contexto em que você o implantará
- Estruturar consultas de upsert para aproveitar as vantagens do mecanismo do DFE
- Criação de gravações eficientes com multi-thread do Gremlin
- Redução de registros com o Creation Time Property
- Ao usar o método datetime( ) para o Groovy Time Data
- Uso da data e hora nativas para dados de data GLV

Testar o código do Gremlin no contexto em que você o implantará

No Gremlin, há várias maneiras de os clientes enviarem consultas ao servidor: usando o WebSocket, o Bytecode GLV ou o console do Gremlin com o uso de scripts baseados em strings.

É importante reconhecer que a execução da consulta do Gremlin pode ser diferente dependendo de como você envia a consulta. Uma consulta que gere um resultado vazio pode ser tratada como bem-sucedida se enviada no modo Bytecode, mas com falha se enviada no modo script. Por exemplo,

se você incluir `next()` em uma consulta no modo script, ela será enviada ao servidor `next()`, mas usando o ByteCode, o cliente geralmente processa a consulta `next()` por conta própria. No primeiro caso, a consulta falhará se nenhum resultado for encontrado, mas no segundo, a consulta será bem-sucedida independentemente de o conjunto de resultados estar vazio ou não.

Se você desenvolver e testar o código em um contexto (por exemplo, o console do Gremlin, que geralmente envia consultas em formato de texto), mas depois implanta o código em um contexto diferente (por exemplo, por meio do driver do Java usando Bytecode), você pode ter problemas com o código apresentando comportamentos diferentes na produção e no ambiente de desenvolvimento.

> **Important**
>
> Teste o código do Gremlin no contexto do GLV em que ele será implantado, para evitar resultados inesperados.

Estruturar consultas de upsert para aproveitar as vantagens do mecanismo do DFE

É possível melhorar significativamente o desempenho das consultas de upsert utilizando o mecanismo do DFE do Neptune da forma mais completa possível.

Criar surtos eficientes com as etapas `mergeV()` e `mergeE()` do Gremlin. explica como estruturar consultas de upsert para usar o mecanismo do DFE da forma mais eficaz possível.

Criação de gravações eficientes com multi-thread do Gremlin

Há algumas diretrizes para carregamento de dados com multi-thread para o Neptune usando o Gremlin.

Se possível, atribua a cada thread um conjunto de vértices ou bordas a serem inseridos ou modificados que não colidam. Por exemplo, o thread abrange o intervalo de IDs de 1 a 50.000, o thread 2 abrange o intervalo de IDs de 50.001 a 100.000, etc. Isso reduz a chance de atingir uma `ConcurrentModificationException`. Para melhor segurança, coloque um bloco `try/catch` em torno de todas as gravações. Se ocorrer uma falha em qualquer uma delas, você poderá repeti-las depois de um breve atraso.

O agrupamento de gravações em lotes de 50 a 100 (vértices ou bordas) geralmente funciona bem. Se você tiver uma grande quantidade de propriedades sendo adicionadas para cada vértice, um

número mais próximo de 50 do que de 100 poderá ser a melhor opção. Alguma experimentação é útil. Portanto, para gravações em lote, você pode usar algo como o seguinte:

```
g.addV('test').property(id,'1').as('a').
  addV('test').property(id,'2').
  addE('friend').to('a').
```

Isso é, então, repetido em cada operação em lote.

O uso de lotes é significativamente mais eficiente do que a adição de um vértice ou borda por Gremlin de ida e volta para o servidor.

Se você estiver usando um cliente de Variante da linguagem Gremlin (GLV), crie um lote de forma programática primeiro criando uma travessia. Em seguida, adicione a ele e, por fim, itere sobre ele; por exemplo:

```
  t.addV('test').property(id,'1').as('a')
  t.addV('test').property(id,'2')
  t.addE('friend').to('a')
  t.iterate()
```

É melhor usar o cliente da variante de linguagem do Gremlin, se possível. No entanto, você pode fazer algo parecido com um cliente que envia consultas como strings de texto concatenando as strings para criar um lote.

Se você estiver usando uma das bibliotecas do cliente de Gremlin em vez de HTTP básico para consultas, os threads deverão compartilhar o mesmo cliente, cluster ou grupo de conexões. Pode ser necessário ajustar as configurações para obter o melhor throughput possível, configurações como o tamanho do grupo de conexões e o número de threads de operador que o cliente do Gremlin usa.

Redução de registros com o Creation Time Property

É possível remover os registros de tabela ao armazenar a hora de criação como uma propriedade em vértices e removê-los periodicamente.

Caso precise armazenar dados para uma vida útil específica e então removê-la do gráfico (tempo de vida do vértice), você pode armazenar uma propriedade time stamp na criação do vértice. Em seguida, você pode periodicamente emitir uma consulta do `drop()` para todos os vértices que foram criados antes de um certo período, por exemplo:

```
g.V().has(“timestamp”, lt(datetime('2018-10-11')))
```

Ao usar o método **`datetime( )`** para o Groovy Time Data

O Neptune oferece o método `datetime` para especificar datas e horas para consultas enviadas na variante Gremlin Groovy. Isso inclui o Gremlin Console, strings de texto usando a API REST HTTP, e qualquer outra serialização que usa o Groovy.

> **Important**
>
> Isso se aplica somente a métodos onde você envia a consulta Gremlin como uma string de texto. Se você estiver usando uma Variante da Linguagem Gremlin, deverá usar as classes e funções de data nativa para o idioma. Para obter mais informações, consulte a próxima sessão, the section called “Data e hora nativas”.
> Começar com o TinkerPop `3.5.2` (introduzido na versão 1.1.1.0 do mecanismo do Neptune), `datetime` é parte integrante do TinkerPop.

Você pode usar o método do `datetime` para armazenar e comparar datas:

```
g.V('3').property('date',datetime('2001-02-08'))
```

```
g.V().has('date',gt(datetime('2000-01-01')))
```

Uso da data e hora nativas para dados de data GLV

Se estiver usando uma Variante da Linguagem Gremlin (GLV, Gremlin Language Variant), você deve usar as classes e funções de data e hora nativas fornecidas pela linguagem de programação para dados de tempo do Gremlin.

As bibliotecas oficiais TinkerPop Java, Node.js (JavaScript), Python, ou .NET são todas bibliotecas Variantes da Linguagem Gremlin.

> **Important**
>
> Isto se aplica somente às bibliotecas Variantes da Linguagem Gremlin (GLV). Se estiver usando um método para enviar a consulta do Gremlin como string de texto, você deverá usar o método `datetime()` fornecido pelo Neptune. Isso inclui o Gremlin Console, strings de

texto usando a API REST HTTP, e qualquer outra serialização que usa o Groovy. Para obter mais informações, consulte a seção anterior, the section called “datetime( )”.

Python

Veja a seguir um exemplo parcial em Python que cria uma propriedade única chamada 'date' para o vértice com um ID do '3'. Isso define o valor a ser uma data gerada usando o método do datetime.now() Python.

```
import datetime

g.V('3').property('date',datetime.datetime.now()).next()
```

Para ver um exemplo completo de como se conectar ao Neptune usando Python, consulte Usar o Python para conectar-se a uma instância de banco de dados do Neptune.

Node.js (JavaScript)

Veja a seguir um exemplo parcial em JavaScript que cria uma propriedade única chamada 'date' para o vértice com um ID do '3'. Isso define o valor a ser uma data gerada usando o construtor do Date() Node.js.

```
g.V('3').property('date', new Date()).next()
```

Para ver um exemplo completo de como se conectar ao Neptune usando Node.js, consulte Usar o Node.js para conectar-se a uma instância de banco de dados do Neptune .

Java

Veja a seguir um exemplo parcial em Java que cria uma propriedade única chamada 'date' para o vértice com um ID do '3'. Isso define o valor a ser uma data gerada usando o construtor do Date() Java.

```
import java.util.date

g.V('3').property('date', new Date()).next();
```

Para ver um exemplo completo de como se conectar ao Neptune usando Java, consulte Usar o cliente do Java para conectar-se a uma instância de banco de dados do Neptune.

.NET (C#)

Veja a seguir um exemplo parcial em C# que cria uma propriedade única chamada '`date`' para o vértice com um ID do '`3`'. Isso define o valor a ser uma data gerada usando a propriedade do *`DateTime.UtcNow`* .NET.

```
Using System;

g.V('3').property('date', DateTime.UtcNow).next()
```

Para ver um exemplo completo de como se conectar ao Neptune usando C#, consulte Usar .NET para conectar-se a uma instância de banco de dados do Neptune.

Práticas recomendadas para usar o cliente Java do Gremlin com o Neptune

Use a versão mais recente compatível do cliente Apache TinkerPop Java

Se você puder, sempre use a versão mais recente do cliente Java Apache TinkerPop Gremlin compatível com a versão do mecanismo que você está usando. As versões mais recentes contêm várias correções de erros que podem melhorar a estabilidade, o desempenho e a usabilidade do cliente.

Consulte Cliente Apache TinkerPop Java Gremlin para obter uma lista das versões do cliente que são compatíveis com várias versões do mecanismo do Neptune.

Reutilizar o objeto de cliente entre vários threads

Reutilize o mesmo objeto de cliente (ou `GraphTraversalSource`) em vários threads. Ou seja, crie uma instância compartilhada de uma classe `org.apache.tinkerpop.gremlin.driver.Client` em seu aplicativo, em vez de fazê-lo em cada thread. O objeto `Client` é seguro para threads, e a sobrecarga de inicializá-lo é considerável.

Isso também se aplica a `GraphTraversalSource`, que cria um objeto `Client` internamente. Por exemplo, o código a seguir cria um novo objeto `Client` a ser instanciado:

```
import static
 org.apache.tinkerpop.gremlin.process.traversal.AnonymousTraversalSource.traversal;
```

```
  /////

GraphTraversalSource traversal = traversal()
                                    .withRemote(DriverRemoteConnection.using(cluster));
```

Criar objetos separados do cliente Java do Gremlin para endpoints de leitura e gravação

É possível aumentar o desempenho ao executar somente gravações no endpoint de gravação e leitura em um ou mais endpoints somente leitura.

```
Client readerClient = Cluster.build("https://reader-endpoint")
          ...
          .connect()

Client writerClient = Cluster.build("https://writer-endpoint")
          ...
          .connect()
```

Adicionar vários endpoints de réplica de leitura a um grupo conexão Java do Gremlin

Ao criar um objeto `Cluster` do Gremlin Java, você pode usar o método `.addContactPoint()` para adicionar várias instâncias de réplica de leitura aos pontos de contato do grupo de conexão.

```
Cluster.Builder readerBuilder = Cluster.build()
          .port(8182)
          .minConnectionPoolSize(…)
          .maxConnectionPoolSize(…)
          ……
          .addContactPoint("reader-endpoint-1")
          .addContactPoint("reader-endpoint-2")
```

Fechar o cliente para evitar o limite de conexões

É importante fechar o cliente quando terminar de usá-lo para garantir que as WebSocket conexões sejam fechadas pelo servidor e que todos os recursos associados às conexões sejam liberados. Isso ocorrerá automaticamente se você fechar o cluster usando `Cluster.close( )`, porque `client.close( )` depois será chamado internamente.

Se o cliente não estiver fechado corretamente, o Neptune encerrará todas as conexões WebSocket ociosas após 20 a 25 minutos. No entanto, se você não fechar explicitamente WebSocket as conexões ao terminar de usá-las e o número de conexões ativas atingir o limite de conexões WebSocket simultâneas, conexões adicionais serão recusadas com um código de 429 erro HTTP. Nesse ponto, você deve reiniciar a instância do Neptune para fechar as conexões.

A orientação de chamar `cluster.close()` não se aplica às funções Java do AWS Lambda . Para obter mais detalhes, consulte Gerenciar conexões do WebSocket do Gremlin em funções do AWS Lambda.

Criar uma conexão após o failover

No caso de failover, o Gremlin Driver pode continuar a se conectar ao gravador antigo porque o nome do DNS do cluster foi resolvido para um endereço IP. Se isso acontecer, crie um novo objeto `Client` após o failover.

Definir **`maxInProcessPerConnection`** e **`maxSimultaneousUsagePerConnection`** como o mesmo valor

Tanto os parâmetros `maxInProcessPerConnection` quanto os `maxSimultaneousUsagePerConnection` parâmetros estão relacionados ao número máximo de consultas simultâneas que você pode enviar em uma única WebSocket conexão. Internamente, esses parâmetros são correlacionados, e a modificação de um deles sem o outro pode fazer com que um cliente receba um tempo limite ao tentar obter uma conexão do grupo de conexões do cliente.

Recomendamos manter os valores em andamento e de uso simultâneo mínimos padrão e definir `maxInProcessPerConnection` e `maxSimultaneousUsagePerConnection` como o mesmo valor.

O valor para definir esses parâmetros é uma função de complexidade de consulta e o modelo de dados. Um caso de uso em que a consulta retorna uma grande quantidade de dados exigiria mais largura de banda de conexão por consulta e, portanto, deve ter valores inferiores para os parâmetros e um valor mais alto para `maxConnectionPoolSize`.

Por outro lado, em um caso em que a consulta retorna uma menor quantidade de dados, `maxInProcessPerConnection` e `maxSimultaneousUsagePerConnection` devem ser definidos como um valor maior que `maxConnectionPoolSize`.

Enviar consultas ao servidor como bytecode em vez de strings

Há vantagens de usar bytecode em vez de string ao enviar consultas:

- Obter a sintaxe de consulta inválida com antecedência: o uso da variante de bytecode permite detectar sintaxe de consulta inválida na fase de compilação. Se você usar a variação com base em string, só detectará a sintaxe inválida quando a consulta for enviada ao servidor e um erro for gerado.
- Evite penalidades de desempenho com base em strings: qualquer envio de consulta com base em string, seja usando WebSockets HTTP, resulta em um vértice separado, o que implica que o objeto Vertex consiste no ID, no rótulo e em todas as propriedades associadas ao vértice (consulte Propriedades dos elementos).

  Isso pode ocasionar computação desnecessária no servidor em casos em que as propriedades não são necessárias. Por exemplo, se o cliente estiver interessado em obter o vértice com o ID "hakuna#1" usando a consulta, `g.V("hakuna#1")`. Se a consulta for enviada como um envio baseado em string, o servidor poderá gastar tempo recuperando o ID, o rótulo e todas as propriedades para esse vértice. Se a consulta for enviada como um envio de bytecode, o servidor apenas gastará tempo recuperando o ID e o rótulo do vértice.

Em outras palavras, em vez de enviar uma consulta da seguinte maneira:

```
 final Cluster cluster = Cluster.build("localhost")
                                .port(8182)
                                .maxInProcessPerConnection(32)
                                .maxSimultaneousUsagePerConnection(32)
                                .serializer(Serializers.GRAPHBINARY_V1D0)
                                .create();

 try {
     final Client client = cluster.connect();
     List<Result> results =
client.submit("g.V().has('name','pumba').out('friendOf').id()").all().get();
     System.out.println(verticesWithNamePumba);
 } finally {
     cluster.close();
 }
```

Em vez disso, envie a consulta usando bytecode, como:

```
 final Cluster cluster = Cluster.build("localhost")
                                .port(8182)
                                .maxInProcessPerConnection(32)
                                .maxSimultaneousUsagePerConnection(32)
                                .serializer(Serializers.GRAPHBINARY_V1D0)
                                .create();

 try {
     final GraphTraversalSource g =
traversal().withRemote(DriverRemoteConnection.using(cluster));
     List<Object> verticesWithNamePumba = g.V().has("name",
"pumba").out("friendOf").id().toList();
     System.out.println(verticesWithNamePumba);
 } finally {
     cluster.close();
 }
```

Sempre consuma completamente o ResultSet ou Iterator retornado por uma consulta

O objeto de cliente sempre deve consumir completamente o `ResultSet` (no caso de envio com base em string) ou o iterador retornado por `GraphTraversal`. Se os resultados da consulta não forem completamente consumidos, o servidor os manterá, aguardando o cliente terminar de consumi-los.

Se o seu aplicativo precisar apenas de um conjunto parcial de resultados, use uma etapa `limit(X)` com sua consulta para restringir o número de resultados que o servidor gera.

Adicionar em massa vértices e bordas em lotes

Cada consulta ao banco de dados do Neptune é executada no escopo de uma única transação, a menos que você use uma sessão. Isso significa que, se você precisa inserir uma grande quantidade de dados usando consultas do gremlin, agrupá-los em um lote de 50-100 melhora o desempenho reduzindo o número de transações criadas para a carga.

Por exemplo, a inclusão de 5 vértices ao banco de dados teria a seguinte aparência:

```
// Create a GraphTraversalSource for the remote connection
final GraphTraversalSource g =
 traversal().withRemote(DriverRemoteConnection.using(cluster));
// Add 5 vertices in a single query
```

```
g.addV("Person").property(T.id, "P1")
 .addV("Person").property(T.id, "P2")
 .addV("Person").property(T.id, "P3")
 .addV("Person").property(T.id, "P4")
 .addV("Person").property(T.id, "P5").iterate();
```

Desativar o armazenamento em cache DNS no Java Virtual Machine

Em um ambiente no qual você deseja fazer o balanceamento de carga das solicitações em várias réplicas de leitura, você precisa desativar o armazenamento em cache DNS no Java Virtual Machine (JVM) e fornecer o endpoint de leitura do Netuno ao criar o cluster. Desabilitar o cache DNS do JVM garante que o DNS seja resolvido novamente para cada nova conexão, de maneira que as solicitações sejam distribuídas em todas as réplicas de leitura. É possível fazer isso no código de inicialização da aplicação com a seguinte linha:

```
java.security.Security.setProperty("networkaddress.cache.ttl", "0");
```

No entanto, uma solução mais completa e robusta para balanceamento de carga é fornecida pelo código do cliente Amazon Gremlin Java on. GitHub O cliente Java do Amazon Gremlin está ciente da topologia do cluster e distribui de forma justa as conexões e solicitações em um conjunto de instâncias no cluster do Neptune. Consulte esta postagem no blog para ver um exemplo da função do Lambda em Java que usa esse cliente.

Opcionalmente, definir tempos limite em um nível por consulta

O Neptune oferece a capacidade de definir um tempo limite para suas consultas usando a opção de grupo de parâmetros `neptune_query_timeout` (consulte Parâmetros). No entanto, a partir da versão 3.3.7 do cliente Java, você também pode substituir o tempo limite global, com o seguinte código:

```
  final Cluster cluster = Cluster.build("localhost")
                                 .port(8182)
                                 .maxInProcessPerConnection(32)
                                 .maxSimultaneousUsagePerConnection(32)
                                 .serializer(Serializers.GRAPHBINARY_V1D0)
                                 .create();

  try {
      final GraphTraversalSource g =
traversal().withRemote(DriverRemoteConnection.using(cluster));
```

```
    List<Object> verticesWithNamePumba = g.with(ARGS_EVAL_TIMEOUT,
500L).V().has("name", "pumba").out("friendOf").id().toList();
    System.out.println(verticesWithNamePumba);
 } finally {
    cluster.close();
 }
```

Ou, para envio de consulta baseada em string, o código teria a seguinte aparência:

```
RequestOptions options = RequestOptions.build().timeout(500).create();
List<Result> result = client.submit("g.V()", options).all().get();
```

> **Note**
>
> Você poderá gerar custos inesperados se definir um valor de tempo limite de consulta muito alto, especialmente em uma instância sem servidor. Sem uma configuração de tempo limite razoável, a consulta poderá continuar sendo executada por muito mais tempo do que o esperado, gerando custos jamais previstos. Esse é particularmente o caso em uma instância sem servidor cuja escala pode ser aumentada verticalmente para um tipo de instância grande e caro durante a execução da consulta.
> É possível evitar despesas inesperadas desse tipo usando um valor de tempo limite de consulta que acomode o tempo de execução esperado e ocasione apenas um tempo limite de execução excepcionalmente longo.

Solução de problemas de `java.util.concurrent.TimeoutException`

O cliente Gremlin Java lança um `java.util.concurrent.TimeoutException` quando uma solicitação do Gremlin expira no próprio cliente enquanto espera que um slot em uma das WebSocket conexões fique disponível. Essa duração do tempo limite é controlada pelo parâmetro configurável `maxWaitForConnection` do lado do cliente.

> **Note**
>
> Como as solicitações que atingem o tempo limite no cliente nunca são enviadas ao servidor, elas não são refletidas em nenhuma das métricas capturadas no servidor, como `GremlinRequestsPerSec`.

Esse tipo de tempo limite geralmente é causado de duas maneiras:

- Na verdade, o servidor atingiu a capacidade máxima. Se for esse o caso, a fila no servidor é preenchida, o que você pode detectar monitorando a métrica MainRequestQueuePending CloudWatch Solicitações. O número de consultas paralelas que o servidor pode processar depende do tamanho da instância.

  Se a métrica `MainRequestQueuePendingRequests` não mostrar um acúmulo de solicitações pendentes no servidor, o servidor poderá lidar com mais solicitações e o tempo limite será causado pelo controle de utilização do lado do cliente.
- Controle de utilização de solicitações pelo cliente. Geralmente, isso pode ser corrigido alterando as configurações do cliente.

  O número máximo de solicitações paralelas que o cliente pode enviar pode ser estimado aproximadamente da seguinte forma:

  ```
  maxParallelQueries = maxConnectionPoolSize * Max( maxSimultaneousUsagePerConnection,
   maxInProcessPerConnection )
  ```

  Enviar mais do que `maxParallelQueries` ao cliente causa exceções `java.util.concurrent.TimeoutException`. Geralmente, é possível corrigir isso de várias maneiras:

  - Aumente a duração do tempo limite da conexão. Se a latência não for essencial para a aplicação, aumente a configuração `maxWaitForConnection` do cliente. O cliente então espera mais antes de atingir o tempo limite o que, por sua vez, pode aumentar a latência.
  - Aumente o máximo de solicitações por conexão. Isso permite que mais solicitações sejam enviadas usando a mesma WebSocket conexão. Faça isso aumentando as configurações `maxSimultaneousUsagePerConnection` e `maxInProcessPerConnection` do cliente. Essas configurações geralmente devem ter o mesmo valor.
  - Aumente o número de conexões no grupo de conexões. Faça isso aumentando a configuração `maxConnectionPoolSize` do cliente. O custo é o aumento do consumo de recursos, porque cada conexão usa memória e um descritor de arquivo do sistema operacional e exige um SSL e um handshake durante a inicialização. WebSocket

Práticas recomendadas para o Neptune ao usar openCypher e Bolt

Siga estas práticas recomendadas ao usar a linguagem de consulta openCypher e o protocolo Bolt com Neptune. Para obter informações sobre como usar o openCypher no Neptune, consulte Acessar o grafo do Neptune com o openCypher.

Preferir direcionar para bordas bidirecionais nas consultas

Quando o Neptune realiza otimizações de consulta, as bordas bidirecionais dificultam a criação de planos de consulta ideais. Planos abaixo do ideal exigem que o mecanismo faça um trabalho desnecessário e ocasionam degradação do desempenho.

Portanto, use bordas direcionadas em vez de bidirecionais sempre que possível. Por exemplo, use:

```
MATCH p=(:airport {code: 'ANC'})-[:route]->(d) RETURN p)
```

em vez de:

```
MATCH p=(:airport {code: 'ANC'})-[:route]-(d) RETURN p)
```

Na verdade, a maioria dos modelos de dados não precisa percorrer bordas nas duas direções, portanto, as consultas podem obter melhorias significativas no desempenho ao passar a usar bordas direcionadas.

Se o modelo de dados exigir percurso por bordas bidirecionais, faça do primeiro nó (lado esquerdo) no padrão `MATCH` o nó com a filtragem mais restritiva.

Veja o exemplo: “Encontre-me em todas as `routes` para e do aeroporto `ANC`”. Escreva essa consulta para começar no aeroporto `ANC`, da seguinte forma:

```
MATCH p=(src:airport {code: 'ANC'})-[:route]-(d) RETURN p
```

O mecanismo pode realizar a quantidade mínima de trabalho para atender à consulta, porque o nó mais restrito é colocado como o primeiro nó (lado esquerdo) no padrão. O mecanismo pode então otimizar a consulta.

Essa é de longe a opção preferencial em comparação a filtrar o aeroporto `ANC` no final do padrão, desta forma:

```
MATCH p=(d)-[:route]-(src:airport {code: 'ANC'}) RETURN p
```

Quando o nó mais restrito não é colocado primeiro no padrão, o mecanismo precisa realizar um trabalho adicional porque não pode otimizar a consulta e precisa realizar pesquisas adicionais para chegar aos resultados.

O Neptune não é compatível com várias consultas simultâneas em uma transação.

Embora o próprio driver do Bolt permita consultas simultâneas em uma transação, o Neptune não é compatível com várias consultas em uma transação executada simultaneamente. Em vez disso, o Neptune exige que várias consultas em uma transação sejam executadas sequencialmente e que os resultados de cada consulta sejam completamente consumidos antes que a próxima consulta seja iniciada.

O exemplo abaixo mostra como usar o Bolt para executar várias consultas sequencialmente em uma transação, para que os resultados de cada uma sejam completamente consumidos antes do início da próxima:

```
final String query = "MATCH (n) RETURN n";

try (Driver driver = getDriver(HOST_BOLT, getDefaultConfig())) {
  try (Session session = driver.session(readSessionConfig)) {
    try (Transaction trx = session.beginTransaction()) {
      final Result res_1 = trx.run(query);
      Assert.assertEquals(10000, res_1.list().size());
      final Result res_2 = trx.run(query);
      Assert.assertEquals(10000, res_2.list().size());
    }
  }
}
```

Criar uma conexão após o failover

No caso de um failover, o driver do Bolt pode continuar se conectando à instância de gravador antiga e não à nova instância ativa, porque o nome do DNS foi resolvido para um endereço IP específico.

Para evitar isso, feche e reconecte o objeto `Driver` após qualquer failover.

Tratamento de conexões para aplicações de longa duração

Ao criar aplicações de longa duração, como aqueles executados em contêineres ou em instâncias do Amazon EC2, instancie um objeto `Driver` uma vez e, depois, reutilize esse objeto durante toda a vida útil da aplicação. O objeto `Driver` é seguro para threads, e a sobrecarga de inicializá-lo é considerável.

Manipulação de conexão para AWS Lambda

Os drivers de parafuso não são recomendados para uso em AWS Lambda funções, devido à sobrecarga de conexão e aos requisitos de gerenciamento. Em vez disso, use o endpoint HTTPS.

Fechar objetos de driver ao concluir

Feche o cliente quando terminar de usá-lo, para que as conexões do Bolt sejam encerradas pelo servidor e todos os recursos associados à conexão sejam liberados. Isso acontecerá automaticamente se você fechar o driver usando `driver.close()`.

Se o driver não for fechado corretamente, o Neptune encerrará todas as conexões inativas do Bolt após vinte minutos ou após dez dias se você estiver usando a autenticação do IAM.

O Neptune não é compatível mais de mil conexões simultâneas do Bolt. Se você não fechar explicitamente as conexões ao terminar de usá-las e o número de conexões ativas atingir o limite de mil, qualquer nova tentativa de conexão falhará.

Usar modos de transação explícitos para leitura e gravação

Ao usar transações com o Neptune e o driver do Bolt, é melhor definir explicitamente o modo de acesso para transações de leitura e gravação como as configurações corretas.

Transações somente leitura

Para transações somente leitura, se você não transmitir a configuração apropriada do modo de acesso ao criar a sessão, o nível de isolamento padrão será usado, que é o isolamento da consulta de mutação. Como resultado, para as transações somente leitura, é importante definir o modo de acesso como `read` explicitamente.

Exemplo de transação de leitura de confirmação automática:

```
SessionConfig sessionConfig = SessionConfig
```

```
  .builder()
  .withFetchSize(1000)
  .withDefaultAccessMode(AccessMode.READ)
  .build();
Session session = driver.session(sessionConfig);
try {
  (Add your application code here)
} catch (final Exception e) {
  throw e;
} finally {
  driver.close()
}
```

Leia o exemplo de transação:

```
Driver driver = GraphDatabase.driver(url, auth, config);
SessionConfig sessionConfig = SessionConfig
  .builder()
  .withDefaultAccessMode(AccessMode.READ)
  .build();
driver.session(sessionConfig).readTransaction(
  new TransactionWork<List<String>>() {
    @Override
    public List<String> execute(org.neo4j.driver.Transaction tx) {
      (Add your application code here)
    }
  }
);
```

Nos dois casos, o isolamento de SNAPSHOT é obtido usando a semântica de transação somente leitura do Neptune.

Como as réplicas de leitura só aceitam consultas somente leitura, qualquer consulta enviada a uma réplica de leitura é executada sob a semântica de isolamento SNAPSHOT.

Não há leituras sujas ou não repetíveis para transações somente leitura.

Transações somente leitura

Para consultas de mutação, há três mecanismos diferentes para criar uma transação de gravação, cada um dos quais é ilustrado abaixo:

Exemplo de transação de gravação implícita:

```
Driver driver = GraphDatabase.driver(url, auth, config);
SessionConfig sessionConfig = SessionConfig
  .builder()
  .withDefaultAccessMode(AccessMode.WRITE)
  .build();
driver.session(sessionConfig).writeTransaction(
  new TransactionWork<List<String>>() {
    @Override
    public List<String> execute(org.neo4j.driver.Transaction tx) {
      (Add your application code here)
    }
  }
);
```

Exemplo de transação de gravação de confirmação automática:

```
SessionConfig sessionConfig = SessionConfig
  .builder()
  .withFetchSize(1000)
  .withDefaultAccessMode(AccessMode.Write)
  .build();
Session session = driver.session(sessionConfig);
try {
  (Add your application code here)
} catch (final Exception e) {
    throw e;
} finally {
    driver.close()
}
```

Exemplo de transação de gravação explícita:

```
Driver driver = GraphDatabase.driver(url, auth, config);
SessionConfig sessionConfig = SessionConfig
  .builder()
  .withFetchSize(1000)
  .withDefaultAccessMode(AccessMode.WRITE)
  .build();
Transaction beginWriteTransaction = driver.session(sessionConfig).beginTransaction();
  (Add your application code here)
beginWriteTransaction.commit();
driver.close();
```

Níveis de isolamento para transações de gravação

- As leituras feitas como parte das consultas de mutação são executadas sob isolamento de transações `READ COMMITTED`.
- Não há leituras sujas para leituras feitas como parte de consultas de mutação.
- Registros e intervalos de registros são bloqueados durante a leitura de uma consulta de mutação.
- Quando um intervalo do índice tiver sido lido por uma transação de mutação, há uma forte garantia de que esse intervalo não será modificado por nenhuma transação simultânea até o final da transação de leitura.

As consultas de mutação não são livres de threads.

Sobre conflitos, consulte Resolução de conflitos usando tempos limite de espera de bloqueio.

Em caso de falha, as consultas de mutação não são repetidas automaticamente.

Lógica de novas tentativas para exceções

Para todas as exceções que permitem uma nova tentativa, geralmente é melhor usar uma estratégia de recuo exponencial e repetição que forneça tempos de espera progressivamente maiores entre as novas tentativas, a fim de lidar melhor com problemas transitórios, como erros `ConcurrentModificationException`. Veja um exemplo de padrão de recuo exponencial e repetição:

```
public static void main() {
  try (Driver driver = getDriver(HOST_BOLT, getDefaultConfig())) {
    retriableOperation(driver, "CREATE (n {prop:'1'})")
        .withRetries(5)
        .withExponentialBackoff(true)
        .maxWaitTimeInMilliSec(500)
        .call();
  }
}

protected RetryableWrapper retriableOperation(final Driver driver, final String query){
  return new RetryableWrapper<Void>() {
    @Override
    public Void submit() {
      log.info("Performing graph Operation in a retry manner......");
      try (Session session = driver.session(writeSessionConfig)) {
```

```
        try (Transaction trx =  session.beginTransaction()) {
            trx.run(query).consume();
            trx.commit();
        }
      }
      return null;
    }

    @Override
    public boolean isRetryable(Exception e) {
      if (isCME(e)) {
        log.debug("Retrying on exception.... {}", e);
        return true;
      }
      return false;
    }

    private boolean isCME(Exception ex) {
      return ex.getMessage().contains("Operation failed due to conflicting concurrent
 operations");
    }
  };
}



/**
 * Wrapper which can retry on certain condition. Client can retry operation using this
 class.
 */
@Log4j2
@Getter
public abstract class RetryableWrapper<T> {

  private long retries = 5;
  private long maxWaitTimeInSec = 1;
  private boolean exponentialBackoff = true;

  /**
   * Override the method with custom implementation, which will be called in retryable
 block.
   */
  public abstract T submit() throws Exception;
```

```
  /**
   * Override with custom logic, on which exception to retry with.
   */
  public abstract boolean isRetryable(final Exception e);

  /**
   * Define the number of retries.
   *
   * @param retries -no of retries.
   */
  public RetryableWrapper<T> withRetries(final long retries) {
    this.retries = retries;
    return this;
  }

  /**
   * Max wait time before making the next call.
   *
   * @param time - max polling interval.
   */
  public RetryableWrapper<T> maxWaitTimeInMilliSec(final long time) {
    this.maxWaitTimeInSec = time;
    return this;
  }

  /**
   * ExponentialBackoff coefficient.
   */
  public RetryableWrapper<T> withExponentialBackoff(final boolean expo) {
    this.exponentialBackoff = expo;
    return this;
  }

  /**
   * Call client method which is wrapped in submit method.
   */
  public T call() throws Exception {
    int count = 0;
    Exception exceptionForMitigationPurpose = null;
    do {
      final long waitTime = exponentialBackoff ? Math.min(getWaitTimeExp(retries),
maxWaitTimeInSec) : 0;
      try {
          return submit();
```

```
      } catch (Exception e) {
        exceptionForMitigationPurpose = e;
        if (isRetryable(e) && count < retries) {
          Thread.sleep(waitTime);
          log.debug("Retrying on exception attempt - {} on exception cause - {}",
 count, e.getMessage());
        } else if (!isRetryable(e)) {
          log.error(e.getMessage());
          throw new RuntimeException(e);
        }
      }
    } while (++count < retries);

    throw new IOException(String.format(
          "Retry was unsuccessful.... attempts %d. Hence throwing exception " + "back
 to the caller...", count),
          exceptionForMitigationPurpose);
  }

  /*
   * Returns the next wait interval, in milliseconds, using an exponential backoff
   * algorithm.
   */
  private long getWaitTimeExp(final long retryCount) {
    if (0 == retryCount) {
      return 0;
    }
    return ((long) Math.pow(2, retryCount) * 100L);
  }
}
```

Defina várias propriedades ao mesmo tempo usando uma única cláusula SET

Em vez de usar várias cláusulas SET para definir propriedades individuais, use um mapa para definir várias propriedades para uma entidade ao mesmo tempo.

Você pode usar:

```
MATCH (n:SomeLabel {`~id`: 'id1'})
SET n += {property1 : 'value1',
property2 : 'value2',
```

```
property3 = 'value3'}
```

Em vez de:

```
MATCH (n:SomeLabel {`~id`: 'id1'})
SET n.property1 = 'value1'
SET n.property2 = 'value2'
SET n.property3 = 'value3'
```

A cláusula SET aceita uma única propriedade ou um mapa. Ao atualizar várias propriedades em uma única entidade, o uso de uma única cláusula SET com um mapa permite que as atualizações sejam realizadas em uma única operação em vez de várias operações, que podem ser executadas com mais eficiência.

Use a cláusula SET para remover várias propriedades de uma só vez

Ao usar a linguagem OpenCypher, REMOVE é usado para remover propriedades de uma entidade. No Neptune, cada propriedade removida requer uma operação separada, adicionando latência de consulta. Em vez disso, você pode usar SET com um mapa para definir todos os valores das propriedades como`null`, o que em Neptune é equivalente a remover propriedades. Neptune terá melhor desempenho quando for necessário remover várias propriedades em uma única entidade.

Use:

```
WITH {prop1: null, prop2: null, prop3: null} as propertiesToRemove
MATCH (n)
SET n += propertiesToRemove
```

Em vez de:

```
MATCH (n)
REMOVE n.prop1, n.prop2, n.prop3
```

Use consultas parametrizadas

É recomendável sempre usar consultas parametrizadas ao fazer consultas usando o OpenCypher. O mecanismo de consulta pode aproveitar consultas parametrizadas repetidas para recursos como cache do plano de consulta, em que a invocação repetida da mesma estrutura parametrizada com parâmetros diferentes pode aproveitar os planos em cache. O plano de consulta gerado para

consultas parametrizadas é armazenado em cache e reutilizado somente quando é concluído em 100 ms e os tipos de parâmetros são NUMBER, BOOLEAN ou STRING.

Use:

```
MATCH (n:foo) WHERE id(n) = $id RETURN n
```

Com parâmetros:

```
parameters={"id": "first"}
parameters={"id": "second"}
parameters={"id": "third"}
```

Em vez de:

```
MATCH (n:foo) WHERE id(n) = "first" RETURN n
MATCH (n:foo) WHERE id(n) = "second" RETURN n
MATCH (n:foo) WHERE id(n) = "third" RETURN n
```

Use mapas achatados em vez de mapas aninhados na cláusula UNWIND

Uma estrutura aninhada profunda pode restringir a capacidade do mecanismo de consulta de gerar um plano de consulta ideal. Para aliviar parcialmente esse problema, os seguintes padrões definidos criarão planos ideais para os seguintes cenários:

- Cenário 1: DESENROLE com uma lista de literais cifrados, que inclui NUMBER, STRING e BOOLEAN.
- Cenário 2: DESENROLE com uma lista de mapas nivelados, que inclui somente literais cifrados (NUMBER, STRING, BOOLEAN) como valores.

Ao escrever uma consulta contendo a cláusula UNWIND, use a recomendação acima para melhorar o desempenho.

Exemplo do cenário 1:

```
UNWIND $ids as x
MATCH(t:ticket {`~id`: x})
```

Com parâmetros:

```
parameters={
  "ids": [1, 2, 3]
}
```

Um exemplo do Cenário 2 é gerar uma lista de nós para CRIAR ou MESCLAR. Em vez de emitir várias declarações, use o padrão a seguir para definir as propriedades como um conjunto de mapas nivelados:

```
UNWIND $props as p
CREATE(t:ticket {title: p.title, severity:p.severity})
```

Com parâmetros:

```
parameters={
  "props": [
    {"title": "food poisoning", "severity": "2"},
    {"title": "Simone is in office", "severity": "3"}
  ]
}
```

Em vez de objetos de nós aninhados, como:

```
UNWIND $nodes as n
CREATE(t:ticket n.properties)
```

Com parâmetros:

```
parameters={
  "nodes": [
    {"id": "ticket1", "properties": {"title": "food poisoning", "severity": "2"}},
    {"id": "ticket2", "properties": {"title": "Simone is in office", "severity": "3"}}
  ]
}
```

Coloque nós mais restritivos no lado esquerdo em expressões de caminho de comprimento variável (VLP)

Nas consultas de caminho de comprimento variável (VLP), o mecanismo de consulta otimiza a avaliação escolhendo iniciar a travessia no lado esquerdo ou direito da expressão. A decisão é

baseada na cardinalidade dos padrões no lado esquerdo e direito. Cardinalidade é o número de nós que correspondem ao padrão especificado.

- Se o padrão correto tiver uma cardinalidade de um, o lado direito será o ponto de partida.
- Se o lado esquerdo e o direito tiverem cardinalidade de um, a expansão é verificada em ambos os lados e começa no lado com a expansão menor. Expansão é o número de bordas de saída ou entrada do nó à esquerda e do nó à direita da expressão VLP. Essa parte da otimização é usada somente se o relacionamento VLP for unidirecional e o tipo de relacionamento for fornecido.
- Caso contrário, o lado esquerdo será o ponto de partida.

Para uma cadeia de expressões VLP, essa otimização só pode ser aplicada à primeira expressão. As outras VLPs são avaliadas começando pelo lado esquerdo. Como exemplo, seja a cardinalidade de (a), (b) uma e a cardinalidade de (c) maior que um.

- `(a)-[*1..]->(c)`: A avaliação começa com (a).
- `(c)-[*1..]->(a)`: A avaliação começa com (a).
- `(a)-[*1..]-(c)`: A avaliação começa com (a).
- `(c)-[*1..]-(a)`: A avaliação começa com (a).

Agora, deixe as bordas de entrada de (a) serem duas, e as bordas de saída de (a) três, as bordas de entrada de (b) quatro e as bordas de saída de (b) cinco.

- `(a)-[*1..]->(b)`: A avaliação começa com (a), pois as bordas de saída de (a) são menores que as bordas de entrada de (b).
- `(a)<-[*1..]-(b)`: A avaliação começa com (a), pois as bordas de entrada de (a) são menores que as bordas de saída de (b).

Como regra geral, coloque o padrão mais restritivo no lado esquerdo de uma expressão VLP.

Evite verificações redundantes de rótulos de nós usando nomes de relacionamento granulares

Ao otimizar o desempenho, o uso de rótulos de relacionamento exclusivos dos padrões de nós permite a remoção da filtragem de rótulos nos nós. Considere um modelo gráfico em que o

relacionamento `likes` é usado apenas para definir um relacionamento entre dois `person` nós. Poderíamos escrever a seguinte consulta para encontrar esse padrão:

```
MATCH (n:person)-[:likes]->(m:person)
RETURN n, m
```

A verificação do `person` rótulo em n e m é redundante, pois definimos que o relacionamento só apareça quando ambos forem do tipo. `person` Para otimizar o desempenho, podemos escrever a consulta da seguinte forma:

```
MATCH (n)-[:likes]->(m)
RETURN n, m
```

Esse padrão também pode ser aplicado quando as propriedades são exclusivas de um único rótulo de nó. Suponha que somente `person` os nós tenham a propriedade`email`, portanto, verificar a correspondência do rótulo do nó `person` é redundante. Escrevendo essa consulta como:

```
MATCH (n:person)
WHERE n.email = 'xxx@gmail.com'
RETURN n
```

É menos eficiente do que escrever essa consulta como:

```
MATCH (n)
WHERE n.email = 'xxx@gmail.com'
RETURN n
```

Você só deve adotar esse padrão quando o desempenho for importante e tiver verificações em seu processo de modelagem para garantir que essas etiquetas de borda não sejam reutilizadas para padrões envolvendo outras etiquetas de nós. Se você introduzir posteriormente uma `email` propriedade em outro rótulo de nó`company`, como, os resultados serão diferentes entre essas duas versões da consulta.

Especifique etiquetas de borda sempre que possível

É recomendável fornecer uma etiqueta de borda sempre que possível ao especificar uma borda em um padrão. Considere o exemplo de consulta a seguir, que é usado para vincular todas as pessoas que moram em uma cidade a todas as pessoas que visitaram essa cidade.

```
MATCH (person)-->(city {country: "US"})-->(anotherPerson)
RETURN person, anotherPerson
```

Se seu modelo gráfico vincular pessoas a nós que não sejam apenas cidades usando vários rótulos de borda, ao não especificar o rótulo final, Neptune precisará avaliar caminhos adicionais que serão descartados posteriormente. Na consulta acima, como uma etiqueta de borda não foi fornecida, o mecanismo trabalha mais primeiro e depois filtra os valores para obter o resultado correto. Uma versão melhor da consulta acima pode ser:

```
MATCH (person)-[:livesIn]->(city {country: "US"})-[:visitedBy]->(anotherPerson)
RETURN person, anotherPerson
```

Isso não só ajuda na avaliação, mas permite que o planejador de consultas crie planos melhores. Você pode até mesmo combinar essa prática recomendada com verificações redundantes de rótulos de nós para remover a verificação de rótulos de cidades e escrever a consulta como:

```
MATCH (person)-[:livesIn]->({country: "US"})-[:visitedBy]->(anotherPerson)
RETURN person, anotherPerson
```

Evite usar a cláusula WITH quando possível

A cláusula WITH no OpenCypher atua como um limite em que tudo antes de ser executado e, em seguida, os valores resultantes são passados para as partes restantes da consulta. A cláusula WITH é necessária quando você precisa de agregação provisória ou deseja limitar o número de resultados, mas, além disso, você deve tentar evitar o uso da cláusula WITH. A orientação geral é remover essas cláusulas WITH simples (sem agregação, ordem por ou limite) para permitir que o planejador de consultas trabalhe em toda a consulta para criar um plano globalmente ideal. Por exemplo, suponha que você tenha escrito uma consulta para retornar todas as pessoas que moram em`India`:

```
MATCH (person)-[:lives_in]->(city)
WITH person, city
MATCH (city)-[:part_of]->(country {name: 'India'})
RETURN collect(person) AS result
```

Na versão acima, a cláusula WITH restringe o posicionamento do padrão `(city)-[:part_of]->(country {name: 'India'})` (que é mais restritivo) anterior. `(person)-[:lives_in]-`

`>(city)` Isso torna o plano abaixo do ideal. Uma otimização dessa consulta seria remover a cláusula WITH e permitir que o planejador calculasse o melhor plano.

```
MATCH (person)-[:lives_in]->(city)
MATCH (city)-[:part_of]->(country {name: 'India'})
RETURN collect(person) AS result
```

Coloque filtros restritivos o mais cedo possível na consulta

Em todos os cenários, a colocação antecipada de filtros na consulta ajuda a reduzir as soluções intermediárias que um plano de consulta deve considerar. Isso significa que menos memória e menos recursos computacionais são necessários para executar a consulta.

O exemplo a seguir ajuda você a entender esses impactos. Suponha que você escreva uma consulta para retornar todas as pessoas que moram em`India`. Uma versão da consulta pode ser:

```
MATCH (n)-[:lives_in]->(city)-[:part_of]->(country)
WITH country, collect(n.firstName + " "  + n.lastName) AS result
WHERE country.name = 'India'
RETURN result
```

A versão acima da consulta não é a melhor maneira de alcançar esse caso de uso. O filtro `country.name = 'India'` aparece posteriormente no padrão de consulta. Primeiro, ele coletará todas as pessoas e onde elas moram, as agrupará por país e, em seguida, filtrará apenas o grupo para`country.name = India`. A maneira ideal de consultar somente pessoas residentes `India` e, em seguida, realizar a agregação de coleta.

```
MATCH (n)-[:lives_in]->(city)-[:part_of]->(country)
WHERE country.name = 'India'
RETURN collect(n.firstName + " "  + n.lastName) AS result
```

Uma regra geral é colocar um filtro o mais rápido possível após a introdução da variável.

Verifique explicitamente se as propriedades existem

Com base na semântica do OpenCypher, quando uma propriedade é acessada, ela é equivalente a uma junção opcional e deve reter todas as linhas, mesmo que a propriedade não exista. Se você souber, com base em seu esquema gráfico, que uma propriedade específica sempre existirá para essa entidade, verificar explicitamente a existência dessa propriedade permite que o mecanismo de consulta crie planos ideais e melhore o desempenho.

Considere um modelo gráfico em que nós do tipo `person` sempre tenham uma propriedade`name`. Em vez de fazer isso:

```
MATCH (n:person)
RETURN n.name
```

Verifique explicitamente a existência da propriedade na consulta com uma verificação IS NOT NULL:

```
MATCH (n:person)
WHERE n.name IS NOT NULL
RETURN n.name
```

Não use o caminho nomeado (a menos que seja necessário)

O caminho nomeado em uma consulta sempre tem um custo adicional, o que pode adicionar penalidades em termos de maior latência e uso de memória. Considere a seguinte consulta:

```
MATCH p = (n)-[:commentedOn]->(m)
WITH p, m, n, n.score + m.score as total
WHERE total > 100
MATCH (m)-[:commentedON]->(o)
WITH p, m, n, distinct(o) as o1
RETURN p, m.name, n.name, o1.name
```

Na consulta acima, supondo que queremos apenas conhecer as propriedades dos nós, o uso do caminho “p” é desnecessário. Ao especificar o caminho nomeado como uma variável, a operação de agregação usando DISTINCT ficará cara em termos de tempo e uso de memória. Uma versão mais otimizada da consulta acima poderia ser:

```
MATCH (n)-[:commentedOn]->(m)
WITH m, n, n.score + m.score as total
WHERE total > 100
MATCH (m)-[:commentedON]->(o)
WITH m, n, distinct(o) as o1
RETURN m.name, n.name, o1.name
```

Evite COLLECT (DISTINCT ())

COLLECT (DISTINCT ()) é usado sempre que uma lista deve ser formada contendo valores distintos. COLLECT é uma função de agregação e o agrupamento é feito com base em chaves adicionais

projetadas na mesma instrução. Quando distinto é usado, a entrada é dividida em vários blocos, onde cada pedaço denota um grupo para redução. O desempenho será afetado à medida que o número de grupos aumentar. Em Neptune, é muito mais eficiente executar DISTINCT antes de realmente coletar/formar a lista. Isso permite que o agrupamento seja feito diretamente nas chaves de agrupamento de todo o bloco.

Considere a seguinte consulta:

```
MATCH (n:Person)-[:commented_on]->(p:Post)
WITH n, collect(distinct(p.post_id)) as post_list
RETURN n, post_list
```

Uma maneira mais ideal de escrever essa consulta é:

```
MATCH (n:Person)-[:commented_on]->(p:Post)
WITH DISTINCT n, p.post_id as postId
WITH n, collect(postId) as post_list
RETURN n, post_list
```

Prefira a função de propriedades em vez da pesquisa de propriedades individuais ao recuperar todos os valores de propriedade

A `properties()` função é usada para retornar um mapa contendo todas as propriedades de uma entidade e é muito mais eficiente do que retornar propriedades individualmente.

Supondo que seus `Person` nós contenham 5 propriedades `firstNamelastName`,,`age`,`dept`, `ecompany`, a seguinte consulta seria preferida:

```
MATCH (n:Person)
WHERE n.dept = 'AWS'
RETURN properties(n) as personDetails
```

Em vez de usar:

```
MATCH (n:Person)
WHERE n.dept = 'AWS'
RETURN n.firstName, n.lastName, n.age, n.dept, n.company

=== OR ===
```

```
MATCH (n:Person)
WHERE n.dept = 'AWS'
RETURN {firstName: n.firstName, lastName: n.lastName, age: n.age,
department: n.dept, company: n.company} as personDetails
```

Execute cálculos estáticos fora da consulta

É recomendável resolver cálculos estáticos (operações matemáticas/de string simples) no lado do cliente. Considere este exemplo em que você deseja encontrar todas as pessoas um ano mais velhas ou menos que o autor:

```
MATCH (m:Message)-[:HAS_CREATOR]->(p:person)
WHERE p.age <= ($age + 1)
RETURN m
```

Aqui, `$age` é injetado na consulta por meio de parâmetros e, em seguida, adicionado a um valor fixo. Esse valor é então comparado comp`.age`. Em vez disso, uma abordagem melhor seria fazer a adição no lado do cliente e passar o valor calculado como um parâmetro $ageplusone. Isso ajuda o mecanismo de consulta a criar planos otimizados e evita a computação estática para cada linha de entrada. Seguindo essas diretrizes, uma versão mais eficiente da consulta seria:

```
MATCH (m:Message)-[:HAS_CREATOR]->(p:person)
WHERE p.age <= $ageplusone
RETURN m
```

Entradas em lote usando UNWIND em vez de declarações individuais

Sempre que a mesma consulta precisar ser executada para entradas diferentes, em vez de executar uma consulta por entrada, seria muito mais eficiente executar uma consulta para um lote de entradas.

Se você quiser mesclar em um conjunto de nós, uma opção é executar uma consulta de mesclagem por entrada:

```
MERGE (n:Person {`~id`: $id})
SET n.name = $name, n.age = $age, n.employer = $employer
```

Com parâmetros:

```
params = {id: '1', name: 'john', age: 25, employer: 'Amazon'}
```

A consulta acima precisa ser executada para cada entrada. Embora essa abordagem funcione, ela pode exigir que muitas consultas sejam executadas para um grande conjunto de entradas. Nesse cenário, o agrupamento em lotes pode ajudar a reduzir o número de consultas executadas no servidor, bem como melhorar a taxa de transferência geral.

Use o seguinte padrão:

```
UNWIND $persons as person
MERGE (n:Person {`~id`: person.id})
SET n += person
```

Com parâmetros:

```
params = {persons: [{id: '1', name: 'john', age: 25, employer: 'Amazon'},
{id: '2', name: 'jack', age: 28, employer: 'Amazon'},
{id: '3', name: 'alice', age: 24, employer: 'Amazon'}...]}
```

Recomenda-se a experimentação com diferentes tamanhos de lote para determinar o que funciona melhor para sua carga de trabalho.

Prefiro usar IDs personalizados para nó/relacionamento

O Neptune permite que os usuários atribuam explicitamente IDs em nós e relacionamentos. O ID deve ser globalmente exclusivo no conjunto de dados e determinístico para ser útil. Uma ID determinística pode ser usada como mecanismo de pesquisa ou filtragem, assim como propriedades; no entanto, usar uma ID é muito mais otimizado do ponto de vista da execução da consulta do que usar propriedades. Há vários benefícios em usar IDs personalizados -

- As propriedades podem ser nulas para uma entidade existente, mas a ID deve existir. Isso permite que o mecanismo de consulta use uma junção otimizada durante a execução.
- Quando consultas de mutação simultâneas são executadas, as chances de exceções de modificação simultânea (CMEs) são reduzidas significativamente quando os IDs são usados para acessar os nós, porque menos bloqueios estão assumindo IDs do que propriedades devido à sua exclusividade imposta.
- O uso de IDs evita a chance de criar dados duplicados, pois o Neptune impõe exclusividade aos IDs, diferentemente das propriedades.

O exemplo de consulta a seguir usa uma ID personalizada:

> **Note**
>
> A propriedade `~id` é usada para especificar o ID, mas `id` é armazenada apenas como qualquer outra propriedade.

```
CREATE (n:Person {`~id`: '1', name: 'alice'})
```

Sem usar uma ID personalizada:

```
CREATE (n:Person {id: '1', name: 'alice'})
```

Se estiver usando o último mecanismo, não há imposição de exclusividade e você poderá executar a consulta posteriormente:

```
CREATE (n:Person {id: '1', name: 'john'})
```

Isso cria um segundo nó com `id=1` nome`john`. Nesse cenário, agora você teria dois nós com`id=1`, cada um com um nome diferente - (alice e john).

Evite fazer **`~id`** cálculos na consulta

Ao usar IDs personalizados nas consultas, sempre realize cálculos estáticos fora das consultas e forneça esses valores nos parâmetros. Quando valores estáticos são fornecidos, o mecanismo é mais capaz de otimizar as pesquisas e evitar a digitalização e a filtragem desses valores.

Se você quiser criar bordas entre os nós existentes no banco de dados, uma opção pode ser:

```
UNWIND $sections as section
MATCH (s:Section {`~id`: 'Sec-' + section.id})
MERGE (s)-[:IS_PART_OF]->(g:Group {`~id`: 'g1'})
```

Com parâmetros:

```
parameters={sections: [{id: '1'}, {id: '2'}]}
```

Na consulta acima, o `id` da seção está sendo computado na consulta. Como a computação é dinâmica, o mecanismo não pode embutir identificações estaticamente e acaba escaneando todos os nós da seção. Em seguida, o mecanismo executa a pós-filtragem dos nós necessários. Isso pode ser caro se houver muitos nós de seção no banco de dados.

A melhor maneira de fazer isso é `Sec-` prefixar os ids que estão sendo passados para o banco de dados:

```
UNWIND $sections as section
MATCH (s:Section {`~id`: section.id})
MERGE (s)-[:IS_PART_OF]->(g:Group {`~id`: 'g1'})
```

Com parâmetros:

```
parameters={sections: [{id: 'Sec-1'}, {id: 'Sec-2'}]}
```

Práticas recomendadas para o Neptune ao usar SPARQL

Siga estas práticas recomendadas ao usar a linguagem de consulta SPARQL com Neptune. Para obter informações sobre como usar o SPARQL no Neptune, consulte Acessar o grafo do Neptune com o SPARQL.

Como consultar todos os gráficos nomeados por padrão

O Amazon Neptune associa cada triplo a um grafo nomeado. O gráfico padrão é definido como a união de todos os gráficos nomeados.

Se você enviar uma consulta do SPARQL sem especificar explicitamente um gráfico por meio da palavra-chave `GRAPH` ou construções como `FROM NAMED`, o Neptune sempre considerará todos os trios em sua instância de banco de dados. Por exemplo, a seguinte consulta gera todos os triplos de um endpoint do SPARQL no Neptune:

```
SELECT * WHERE { ?s ?p ?o }
```

Trios que aparecem em mais de um gráfico são retornados somente uma vez.

Para obter informações sobre a especificação de gráfico padrão, consulte a seção Dataset do RDF da especificação do SPARQL 1.1 Query Language.

Como especificar um nome gráfico para carregamento

O Amazon Neptune associa cada triplo a um grafo nomeado. Se você não especificar um grafo nomeado ao carregar, inserir ou atualizar triplos, o Neptune usará o grafo nomeado fallback definido pelo URI, `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`.

Se você estiver usando o carregador em massa do Neptune, será possível especificar um grafo nomeado para usar todos os triplos (ou quadrantes com a quarta posição em branco) usando o parâmetro `parserConfiguration: namedGraphUri`. Para obter informações sobre a sintaxe do comando `Load` do carregador do Neptune, consulte the section called “Comando Loader”.

Como escolher entre FILTER, FILTER...IN e VALUES em suas consultas

Existem três maneiras básicas para injetar valores em consultas SPARQL: `FILTER`, `FILTER...IN` e `VALUES`.

Por exemplo, suponha que você queira pesquisar os amigos de várias pessoas em uma única consulta. Usando `FILTER`, você pode estruturar a consulta da seguinte forma:

```
PREFIX ex: <https://www.example.com/>
PREFIX foaf : <http://xmlns.com/foaf/0.1/>

SELECT ?s ?o
WHERE {?s foaf:knows ?o. FILTER (?s = ex:person1 || ?s = ex:person2)}
```

Isso retorna todos os triplos no gráfico que têm `?s` vinculado a `ex:person1` ou a `ex:person2` e uma borda de saída chamada `foaf:knows`.

Você também pode criar uma consulta usando `FILTER...IN` que retorna resultados equivalentes:

```
PREFIX ex: <https://www.example.com/>
PREFIX foaf : <http://xmlns.com/foaf/0.1/>

SELECT ?s ?o
WHERE {?s foaf:knows ?o. FILTER (?s IN (ex:person1, ex:person2))}
```

Você também pode criar uma consulta usando `VALUES` que, nesse caso, também retorna resultados equivalentes:

```
PREFIX ex: <https://www.example.com/>
PREFIX foaf : <http://xmlns.com/foaf/0.1/>
```

```
SELECT ?s ?o
WHERE {?s foaf:knows ?o. VALUES ?s {ex:person1 ex:person2}}
```

Embora em muitos casos essas consultas sejam semanticamente equivalentes, há alguns casos em que as duas variantes `FILTER` diferem da variante `VALUES`:

- O primeiro caso é quando você injeta valores duplicados, como injetar a mesma pessoa duas vezes. Nesse caso, a consulta `VALUES` inclui as duplicatas no resultado. Você pode eliminar essas duplicações explicitamente adicionando um `DISTINCT` à cláusula `SELECT`. No entanto, pode haver situações em que você realmente quer duplicações nos resultados da consulta de injeção de valor redundante.

  No entanto, as versões `FILTER` e `FILTER...IN` extraem o valor somente uma vez quando o mesmo valor é exibido várias vezes.
- O segundo caso está relacionado ao fato de que `VALUES` sempre realiza uma correspondência exata, enquanto `FILTER` pode aplicar uma promoção de tipo e fazer correspondência difusa em alguns casos.

  Por exemplo, quando você inclui um literal, como `"2.0"^^xsd:float` na cláusula de valores, uma consulta `VALUES` corresponde exatamente a essa literal, incluindo valor de literal e tipo de dados.

  Por outro lado, `FILTER` produz uma correspondência difusa para esses literais numéricos. As correspondências podem incluir literais com o mesmo valor, mas com diferentes tipos de dados numéricos, como `xsd:double`.

  > **Note**
  >
  > Não há diferença entre o comportamento de `FILTER` e de `VALUES` ao enumerar literais de strings ou URIs.

As diferenças entre `FILTER` e `VALUES` podem afetar a otimização e a estratégia de avaliação de consulta resultante. A não ser que seu caso de uso precise de correspondência difusa, recomendamos usar `VALUES` porque ele evita olhar para casos especiais relacionados a conversão do tipo. Como resultado, `VALUES` muitas vezes produz uma consulta mais eficiente cuja execução é mais rápida e mais econômica.

Limites do Amazon Neptune

Regiões

O Amazon Neptune está disponível nas seguintes regiões: AWS

- Leste dos EUA (Norte da Virgínia): `us-east-1`
- Leste dos EUA (Ohio): `us-east-2`
- Oeste dos EUA (N. da Califórnia): `us-west-1`
- Oeste dos EUA (Oregon): `us-west-2`
- Canadá (Central): `ca-central-1`
- América do Sul (São Paulo): `sa-east-1`
- Europa (Estocolmo): `eu-north-1`
- Europa (Irlanda): `eu-west-1`
- Europa (Londres): `eu-west-2`
- Europa (Paris): `eu-west-3`
- Europa (Frankfurt): `eu-central-1`
- Oriente Médio (Bahrein): `me-south-1`
- Oriente Médio (Emirados Árabes Unidos): `me-central-1`
- Israel (Tel Aviv): `il-central-1`
- África (Cidade do Cabo): `af-south-1`
- Ásia-Pacífico (Hong Kong): `ap-east-1`
- Ásia-Pacífico (Tóquio): `ap-northeast-1`
- Ásia-Pacífico (Seul): `ap-northeast-2`
- Ásia-Pacífico (Osaka): `ap-northeast-3`
- Ásia-Pacífico (Singapura): `ap-southeast-1`
- Ásia-Pacífico (Sydney): `ap-southeast-2`
- Ásia-Pacífico (Mumbai): `ap-south-1`
- China (Pequim): `cn-north-1`
- China (Ningxia): `cn-northwest-1`
- AWS GovCloud (Oeste dos EUA): `us-gov-west-1`

- AWS GovCloud (Leste dos EUA): `us-gov-east-1`

Diferenças nas regiões da China

Como acontece com muitos AWS serviços, o Amazon Neptune opera de forma um pouco diferente na China (Pequim) e na China (Ningxia) do que em outras regiões. AWS

Por exemplo, quando o Neptune ML usa o Amazon API Gateway para criar o serviço de exportação, a autenticação do IAM é habilitada por padrão. Nas regiões da China, o processo de alteração dessa opção é um pouco diferente do que em outras regiões.

Essas e outras diferenças são explicadas aqui.

Tamanho máximo dos volumes do cluster de armazenamento

Um volume de cluster Neptune pode crescer até um tamanho máximo de 128 tebibytes (TiB) em todas as regiões suportadas, exceto na China e GovCloud, onde o limite é 64 TiB. Isso vale para todas as versões do mecanismo, a partir de Versão: 1.0.2.2 (09/03/2020). Consulte Armazenamento, confiabilidade e disponibilidade do Amazon Neptune..

Tamanhos de instâncias de banco de dados compatíveis

O Neptune oferece suporte a diferentes classes de instância de banco de dados em diferentes regiões. AWS Para descobrir quais classes têm suporte em uma região, consulte Definição de preço do Amazon Neptune e selecione a região de interesse.

Limites para cada AWS conta

Para determinados recursos de gerenciamento, o Amazon Neptune usa a tecnologia operacional que é compartilhada com o Amazon Relational Database Service (Amazon RDS).

Cada AWS conta tem limites para cada região no número de recursos do Amazon Neptune e do Amazon RDS que você pode criar. Esses recursos incluem instâncias de banco de dados e clusters de banco de dados.

Depois de atingir o limite de um recurso, as chamadas adicionais para criá-lo falham, com uma exceção.

Para obter uma lista dos limites compartilhados entre o Amazon Neptune e o Amazon RDS, consulte Limits in Amazon RDS no Guia do usuário do Amazon RDS.

A conexão com o Neptune requer uma VPC

O Amazon Neptune é uma nuvem privada virtual (VPC): apenas de serviço.

Além disso, as instâncias não permitem acesso de fora da VPC.

O Neptune exige SSL

A partir da versão `1.0.4.0` do mecanismo, o Amazon Neptune só permite conexões Secure Sockets Layer (SSL) por meio de HTTPS para qualquer instância ou endpoint de cluster.

O Neptune exige TLS versão 1.2, usando os seguintes pacotes de criptografia forte:

- `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA`

Zonas de disponibilidade e grupos de sub-redes de banco de dados

O Amazon Neptune exige um grupo de sub-redes de banco de dados para cada cluster que tenha pelo menos duas zonas de disponibilidade (AZs) compatíveis.

Recomendamos usar três ou mais sub-redes em diferentes zonas de disponibilidade.

Carga útil máxima da solicitação HTTP (150 MB)

As solicitações HTTP Gremlin e SPARQL devem ter menos de 150 MB no total. Se uma solicitação exceder esse tamanho, o Neptune retornará `HTTP 400: BadRequestException`.

Esse limite não se aplica às conexões do Gremlin WebSockets .

Diferenças de implementação do Gremlin

A implementação do Gremlin no Amazon Neptune tem detalhes de implementação específicos que podem ser diferentes de outras implementações do Gremlin.

Para ter mais informações, consulte Conformidade com os padrões do Gremlin no Amazon Neptune.

O Neptune não é compatível com caracteres nulos em dados de string

O Neptune não é compatível com caracteres nulos em strings. Isso vale para dados de grafos de propriedades para Gremlin e openCypher e para dados RDF/SPARQL.

SPARQL UPDATE LOAD do URI

O `SPARQL UPDATE LOAD` do URI funciona somente com recursos que estão na mesma VPC.

Isso inclui URLs do Amazon S3 na mesma região que o cluster com uma endpoint da VPC do Amazon S3 criado.

O URL do Amazon S3 deve ser HTTPS e qualquer autenticação deve ser incluída no URL. Para obter mais informações, consulte Authenticating Requests: Using Query Parameters na Referência de API do Amazon Simple Storage Service.

Para obter informações sobre como criar um VPC endpoint, consulte Criar o endpoint da VPC do Amazon S3.

Se for necessário carregar dados de um arquivo, recomendamos usar a API do carregador do Amazon Neptune. Para ter mais informações, consulte Usar o carregador em massa do Amazon Neptune para ingerir dados.

> **Note**
>
> API de carregador do Amazon Neptune é não ACID.

Autenticação e controle de acesso do IAM

Nas versões do mecanismo do Neptune anteriores à versão 1.2.0.0, a autenticação e o controle de acesso do IAM são compatíveis somente em nível de cluster de banco de dados. No entanto, do lançamento da `1.2.0.0` em diante, você pode controlar o acesso baseado em consultas em um nível mais detalhado usando chaves de condição nas políticas do IAM. Para obter mais informações, consulte Usar ações de consulta nas declarações de política de acesso a dados do Neptune e Visão geral do AWS Identity and Access Management (IAM) no Amazon Neptune.

O console do Amazon Neptune exige permissões. NeptuneReadOnlyAccess Você pode restringir o acesso aos usuários do IAM revogando esse acesso. Para obter mais informações, consulte AWS políticas gerenciadas (predefinidas) para o Amazon Neptune.

O Amazon Neptune não é compatível com o controle de acesso com base em nome de usuário/senha.

WebSocket conexões simultâneas e tempo máximo de conexão

Há um limite para o número de WebSocket conexões simultâneas por instância de banco de dados Neptune. Quando esse limite é atingido, o Neptune limita qualquer solicitação para abrir uma WebSocket nova conexão para evitar o uso de toda a memória de pilha alocada.

Para todos os tipos de instâncias maiores compatíveis com o Neptune e todas as instâncias sem servidor, o número máximo de conexões simultâneas WebSocket é 32K (32.768).

O máximo de WebSocket conexões simultâneas para tipos de instâncias menores está listado na tabela abaixo:

Tipo de instância	Máximo de conexões simultâneas WebSocket
db.t3.medium	512
db.t4g.medium	512
db.r5.large	2.048
db.r5d.large	2.048
db.r5.xlarge	4.096

Tipo de instância	Máximo de conexões simultâneas WebSocket
db.r5.2xlarge	8,192
db.r5d.2xlarge	8,192
db.r5.4xlarge	16.384
db.r5d.4xlarge	16.384
db.r6g.large	2.048
db.r6gd.large	2.048
db.r6g.xlarge	4.096
db.r6gd.xlarge	4.096
db.r6g.2xlarge	8,192
db.r6gd.2xlarge	8,192
db.r6g.4xlarge	16.384
db.r6gd.4xlarge	16.384
db.x2g.large	2.048
db.x2gd.large	2.048
db.x2g.xlarge	4.096
db.x2gd.xlarge	4.096
db.x2iedn.xlarge	4.096
db.x2g.2xlarge	8,192
db.x2gd.2xlarge	8,192
db.x2g.4xlarge	16.384

Tipo de instância	Máximo de conexões simultâneas WebSocket
db.x2gd.4xlarge	16.384
db.x2iedn.2xlarge	16.384
db.x2iezn.2xlarge	16.384
tecnologia sem servidor	32.768
(outros tipos de instâncias grandes)	32.768

> **Note**
>
> A partir da versão 1.1.0.0 do mecanismo do Neptune, o Neptune não é mais compatível com tipos de instância R4.

Quando um cliente fechar corretamente uma conexão, isso será refletido imediatamente na contagem de conexões abertas.

Se o cliente não fechar uma conexão, a conexão poderá ser fechada automaticamente após um tempo limite de inatividade de 20 a 25 minutos (o tempo limite de inatividade é o tempo decorrido desde que a última mensagem foi recebida do cliente). No entanto, desde que o tempo limite de inatividade não seja atingido, o Neptune mantém a conexão aberta indefinidamente.

Quando a autenticação do IAM é ativada, uma WebSocket conexão é sempre desconectada alguns minutos a mais de 10 dias após ter sido estabelecida, caso ainda não tenha sido fechada até lá.

Limites em propriedades e rótulos

Não há limite para o número de vértices e bordas ou quadrantes RDF que você pode ter em um gráfico.

Também não há limite para o número de propriedades ou rótulos que qualquer vértice ou borda pode ter.

Há um limite de tamanho de 55 MB no tamanho de uma propriedade ou um rótulo individual. Em termos de RDF, isso significa que o valor em qualquer coluna (S, P, O ou G) de um quadrante RDF não pode exceder 55 MB.

Se precisar associar um objeto maior, como uma imagem, a um vértice ou um nó ao grafo, você poderá armazená-lo como um arquivo no Amazon S3 e usar o caminho do Amazon S3 como a propriedade ou o rótulo.

Limites que afetam o carregador em massa do Neptune

Não é possível colocar em fila mais de 64 trabalhos de carregamento em massa do Neptune por vez.

O Neptune somente mantém o controle dos 1.024 trabalhos de carregamento em massa mais recentes.

O Neptune armazena apenas os últimos dez mil detalhes de erro por trabalho.

Trabalhar com outros serviços da AWS

É possível usar o Amazon Neptune em conjunto com muitos outros serviços da AWS.

Integrações do Neptune a outros serviços

- AWS Glue: AWS Glue é um serviço de integração de dados sem servidor que ajuda a realizar trabalhos de extração, transformação e carregamento (ETL) nos dados.

  O Neptune fornece uma biblioteca de código aberto, neptune-python-utilities, que simplifica o uso do Python e do Gremlin em um trabalho do Glue. O Neo4j Spark Connector também é compatível com a execução de trabalhos do Scala e do openCypher Glue.
- Amazon SageMaker: o Amazon SageMaker é uma plataforma de machine learning completa para criar, treinar e implantar modelos de machine learning de alta qualidade.

  O Neptune se integra ao SageMaker de duas formas principais:

  - O Neptune fornece um pacote Python de código aberto para cadernos Jupyter, que pode ser encontrado no projeto de bloco de anotações Neptune no GitHub. Este pacote contém um conjunto de magias do Jupyter, blocos de anotações de tutoriais e exemplos de código que fornecem um ambiente de codificação interativo onde você pode saber mais sobre a tecnologia de grafos e o Neptune. O Neptune fornece um ambiente totalmente gerenciado para cadernos Jupyter hospedados pelo SageMaker e se vincula automaticamente aos blocos de anotações no projeto de blocos de anotações de grafos Neptune de código aberto.
  - O atributo Neptune ML possibilita criar e treinar modelos úteis de machine learning úteis em grafos grandes em horas e não em semanas. Para isso, o Neptune ML usa a tecnologia de rede neural de grafos (GNN) desenvolvida pelo Amazon SageMaker e pela Deep Graph Library (DGL).
- AWS Lambda: as funções do AWS Lambda têm muitos usos nas aplicações Neptune.

  Para obter informações sobre como usar as funções do Lambda com qualquer um dos drivers e variantes de linguagem populares do Gremlin, bem como exemplos específicos de funções do Lambda escritas em Java, JavaScript e Python, consulte Usar funções do AWS Lambda no Amazon Neptune.
- Amazon Athena: o Amazon Athena é um serviço de consultas interativas que facilita a análise de dados no Amazon Simple Storage Service e outras fontes de dados federados usando SQL padrão.

O Neptune fornece um conector para o Athena que permite ao Athena se comunicar com os dados armazenados no Neptune.

- AWS Database Migration Service (AWS DMS): o AWS Database Migration Service é um serviço Web da AWS que você pode usar para migrar dados de um banco de dados para outro.

  O AWS DMS pode carregar dados no Neptune de bancos de dados de origem compatíveis de forma rápida e segura. O banco de dados de origem permanece totalmente operacional durante a migração, o que minimiza o tempo de inatividade de aplicativos que dependem dele.
- AWS Backup: o AWS Backup é um serviço de backup totalmente gerenciado que facilita a centralização e a automação do backup de dados entre todos os serviços da AWS na nuvem e on-premises.

  O AWS Backup permite criar snapshots periódicos automatizados dos clusters do Neptune usando a política centralizada de proteção de dados em todos os serviços da AWS compatíveis com banco de dados, armazenamento e computação.
- AWS SDK para pandas: o AWS SDK para pandas (anteriormente conhecido como AWS Data Wrangler ou `awswrangler`) é uma iniciativa python de código aberto do AWS Professional Service que amplia o poder da biblioteca de análise de dados `pandas` Python para AWS, conectando `DataFrames` e mais de 30 serviços relacionados a dados da AWS, incluindo o Neptune.

  Além do SDK, há também um tutorial sobre como usá-lo com o Neptune e vários exemplos de blocos de anotações Neptune, como Fraud Ring Detection, Synthetic Identity Detection e Logistics Analysis.
- Driver JDBC: o driver Neptune JDBC é compatível com consultas do openCypher, do Gremlin, do SQL-Gremlin e do SPARQL.

  A conectividade JDBC facilita a conexão com o Neptune com ferramentas de business intelligence (BI), como o Tableau.

Ferramentas e utilitários do Neptune

O Amazon Neptune oferece várias ferramentas e utilitários que podem simplificar e automatizar o trabalho com um grafo. Estes são alguns exemplos:

Ferramentas do Amazon Neptune

- Utilitário do Amazon Neptune para GraphQL: o utilitário do Amazon Neptune para GraphQL é uma ferramenta de linha de comando Node.js de código aberto que pode ajudar a criar e manter uma API GraphQL para um banco de dados de grafos de propriedades do Neptune. É uma forma sem código de criar um resolvedor GraphQL para consultas do GraphQL que tenham um número variável de parâmetros de entrada e geram um número variável de campos aninhados.

Utilitário Amazon Neptune para GraphQL

O utilitário Amazon Neptune para GraphQL é uma ferramenta de linha de comando Node.js de código aberto que pode ajudar a criar e manter uma API GraphQL para um banco de dados de grafos de propriedades do Neptune (ele ainda não funciona com dados RDF). É uma forma sem código de criar um resolvedor GraphQL para consultas do GraphQL que tenham um número variável de parâmetros de entrada e geram um número variável de campos aninhados.

Ele foi lançado como um projeto de código aberto localizado em https://github.com/aws/ amazon-neptune-for-graphql.

É possível instalar o utilitário usando o NPM desta forma (consulte Instalação e configuração para obter detalhes):

```
npm i @aws/neptune-for-graphql -g
```

O utilitário pode descobrir o esquema de grafos de um grafo de propriedades do Neptune existente, incluindo nós, bordas, propriedades e cardinalidade de bordas. Depois, ele gera um esquema GraphQL com as diretivas necessárias para associar os tipos do GraphQL aos nós e às bordas do banco de dados, além de gerar automaticamente o código do resolvedor. O código do resolvedor foi projetado para minimizar a latência gerando somente os dados solicitados pela consulta do GraphQL.

Você também pode começar com um esquema do GraphQL existente e um banco de dados Neptune vazio e deixar que o utilitário deduza as diretivas necessárias para associar esse esquema do

GraphQL aos nós e às bordas dos dados a serem carregados no banco de dados. Ou você pode começar com um esquema e diretivas do GraphQL que você já criou ou modificou.

O utilitário é capaz de criar todos os AWS recursos necessários para seu pipeline, incluindo a AWS AppSync API, as funções do IAM, a fonte de dados, o esquema e o resolvedor, além da função AWS Lambda que consulta Neptune.

> **Note**
>
> Os exemplos de linha de comando aqui pressupõem um console do Linux. Se você estiver usando o Windows, substitua as barras invertidas ("\") no final das linhas por acentos circunflexos ("^").

Tópicos

Instalar e configurar o utilitário Amazon Neptune para GraphQL

Se você for usar o utilitário com um banco de dados Neptune existente, precisará que ele possa se conectar ao endpoint do banco de dados. Por padrão, um banco de dados Neptune só pode ser acessado de dentro da VPC em que está localizado.

Como o utilitário é uma ferramenta de linha de comando do Node.js, você deve ter o Node.js (versão 18 ou superior) instalado para que o utilitário seja executado. Para instalar o Node.js em uma instância do EC2 na mesma VPC do banco de dados Neptune, siga as instruções aqui. O tamanho mínimo da instância para executar o utilitário é t2.micro. Durante a criação da instância, selecione a VPC do banco de dados Neptune no menu suspenso Grupos de segurança comuns.

No entanto, a partir da versão 1.2.0.0 do mecanismo, você pode criar um endpoint público para seu banco de dados Neptune que possa ser acessado fora da VPC. Se você criou um endpoint público, poderá instalar o Node.js e o utilitário na máquina local. Para instalar o Node.js no macOS ou no Windows, acesse o site do Node.js para baixar o instalador.

Para instalar o utilitário em uma instância do EC2 ou na máquina local, use o NPM:

```
npm install aws-neptune-for-graphql -g
```

Depois, você pode executar o comando help do utilitário para conferir se ele foi instalado corretamente:

```
neptune-for-graphql --help
```

Convém instalar o AWS CLI para gerenciar recursos da AWS.

Verificar dados em um banco de dados Neptune existente

Você conheça ou não o GraphQL, o comando abaixo é a maneira mais rápida de criar uma API do GraphQL. Isso pressupõe que você tenha instalado e configurado o utilitário Neptune para GraphQL conforme descrito na seção de instalação, para que ele esteja conectado ao endpoint do banco de dados Neptune.

```
neptune-for-graphql \
  --input-graphdb-schema-neptune-endpoint (your neptune database endpoint):(port
 number) \
  --create-update-aws-pipeline \
  --create-update-aws-pipeline-name (your new GraphQL API name) \
  --output-resolver-query-https
```

O utilitário analisa o banco de dados para descobrir o esquema dos nós, das bordas e das propriedades nele. Com base nesse esquema, ele infere um esquema GraphQL com consultas e mutações associadas. Em seguida, ele cria uma API AppSync GraphQL e os AWS recursos necessários para usá-la. Esses recursos incluem dois perfis do IAM e uma função do Lambda contendo o código do resolvedor GraphQL.

Quando o utilitário terminar, você encontrará uma nova API do GraphQL no AppSync console com o nome que você atribuiu no comando. Para testá-lo, use a opção AppSync Consultas no menu.

Se você executar o mesmo comando novamente após adicionar mais dados ao banco de dados, atualizará a AppSync API e o código Lambda adequadamente.

Para liberar todos os recursos associados ao comando, execute:

```
neptune-for-graphql \
```

```
  --remove-aws-pipeline-name (your new GraphQL API name from above)
```

Iniciar com um esquema GraphQL sem diretivas

Você pode começar com um banco de dados Neptune vazio e usar um esquema GraphQL sem diretivas para criar os dados e consultá-los. O comando abaixo cria automaticamente recursos AWS para fazer o seguinte:

```
neptune-for-graphql \
  --input-schema-file (your GraphQL schema file)
  --create-update-aws-pipeline \
  --create-update-aws-pipeline-name (name for your new GraphQL API) \
  --create-update-aws-pipeline-neptune-endpoint (your Neptune database endpoint):(port
 number) \
  --output-resolver-query-https
```

O arquivo do esquema GraphQL deve incluir os tipos de esquema GraphQL, conforme mostrado no exemplo TODO abaixo. O utilitário analisa seu esquema e cria uma versão estendida com base em seus tipos. Ele adiciona consultas e mutações para os nós armazenados no banco de dados de grafos e, se seu esquema tiver tipos aninhados, ele adicionará relacionamentos entre os tipos armazenados como bordas no banco de dados.

O utilitário cria uma API AppSync GraphQL e todos os AWS recursos necessários. Eles incluem dois perfis do IAM e uma função do Lambda contendo o código do resolvedor GraphQL. Quando o comando for concluído, você poderá encontrar uma nova API do GraphQL com o nome especificado no AppSync console. Para testá-lo, use Consultas no AppSync menu.

O exemplo a seguir ilustra como isso funciona:

Exemplo de Todo, partindo de um esquema GraphQL sem diretivas

Neste exemplo, partimos de um esquema Todo GraphQL sem diretivas, que você pode encontrar no diretório *???samples???*. Ele inclui estes dois tipos:

```
type Todo {
  name: String
  description: String
  priority: Int
  status: String
  comments: [Comment]
}
```

```
type Comment {
  content: String
}
```

Esse comando processa o esquema Todo e um endpoint de um banco de dados Neptune vazio para criar uma API GraphQL em: AWS AppSync

```
neptune-for-graphql /
  --input-schema-file ./samples/todo.schema.graphql \
  --create-update-aws-pipeline \
  --create-update-aws-pipeline-name TodoExample \
  --create-update-aws-pipeline-neptune-endpoint (empty Neptune database endpoint):(port
 number) \
  --output-resolver-query-https
```

O utilitário cria um novo arquivo na pasta de saída chamada`TodoExample.source.graphql`, e a API GraphQL em. AppSync O utilitário infere o seguinte:

- No tipo Todo, ele adicionou `@relationship` um novo CommentEdge tipo. Isso instrui o resolvedor a conectar Todo ao Comment usando uma borda de banco de dados gráfica chamada CommentEdge.
- Ele adicionou uma nova entrada chamada TodoInput para ajudar nas consultas e mutações.
- Ele adicionou duas consultas para cada tipo (Todo, Comment): uma para recuperar um único tipo usando um `id` ou qualquer um dos campos de tipo listados na entrada e outra para recuperar vários valores, filtrados usando a entrada desse tipo.
- Ele adicionou três mutações para cada tipo: criar, atualizar e excluir. O tipo a ser excluído é especificado usando um `id` ou a entrada desse tipo. Essas mutações afetam os dados armazenados no banco de dados Neptune.
- Ele adicionou duas mutações para conexões: conectar e excluir. Eles tomam como entrada os IDs dos nós entre os vértices usados pelo Neptune e a conexão são bordas no banco de dados.

O resolvedor reconhece as consultas e as mutações pelos nomes, mas você pode personalizá-las conforme mostrado abaixo.

Aqui está o conteúdo do arquivo `TodoExample.source.graphql`:

```
type Todo {
```

```
  _id: ID! @id
  name: String
  description: String
  priority: Int
  status: String
  comments(filter: CommentInput, options: Options): [Comment] @relationship(type:
 "CommentEdge", direction: OUT)
  bestComment: Comment @relationship(type: "CommentEdge", direction: OUT)
  commentEdge: CommentEdge
}

type Comment {
  _id: ID! @id
  content: String
}

input Options {
  limit: Int
}

input TodoInput {
  _id: ID @id
  name: String
  description: String
  priority: Int
  status: String
}

type CommentEdge {
  _id: ID! @id
}

input CommentInput {
  _id: ID @id
  content: String
}

input Options {
  limit: Int
}

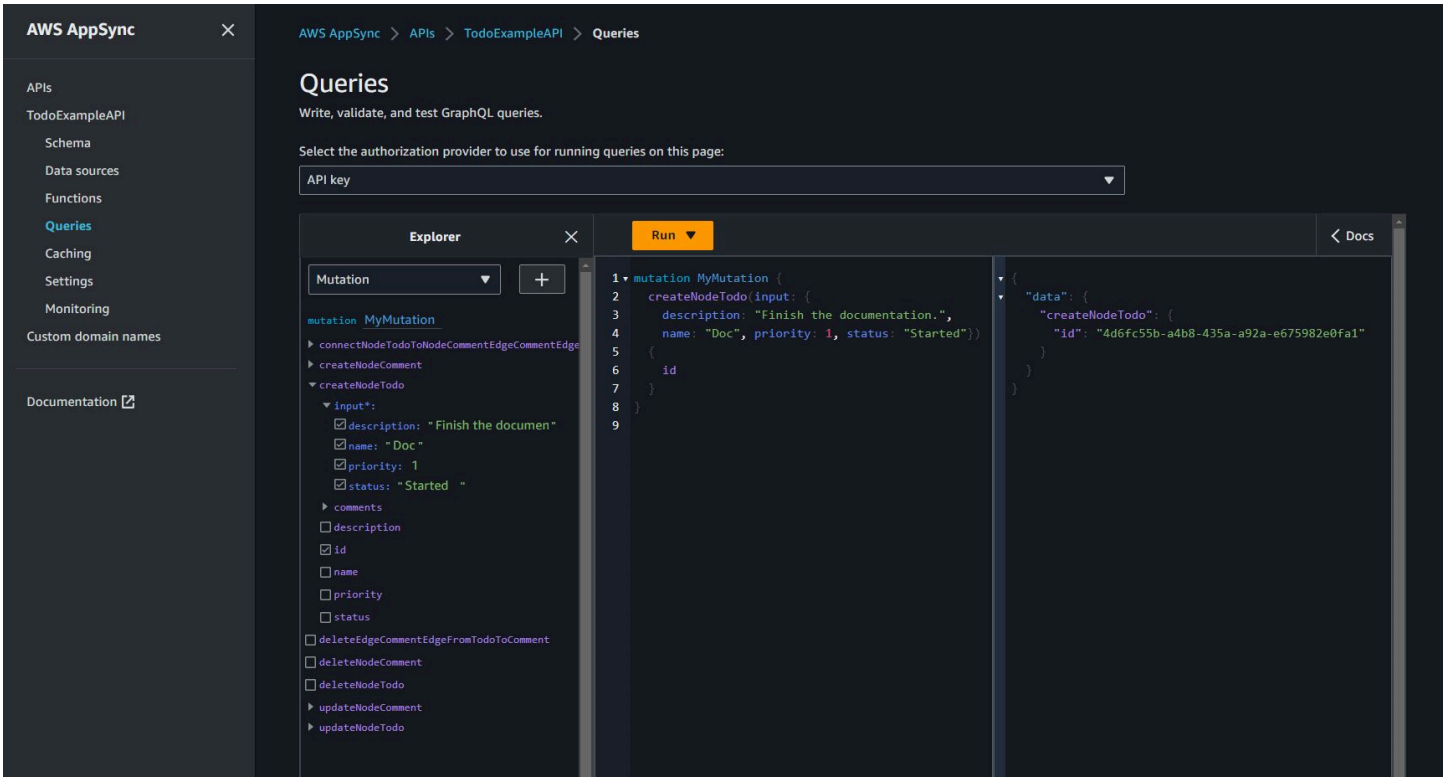
type Query {
  getNodeTodo(filter: TodoInput, options: Options): Todo
  getNodeTodos(filter: TodoInput): [Todo]
```

```
  getNodeComment(filter: CommentInput, options: Options): Comment
  getNodeComments(filter: CommentInput): [Comment]
}

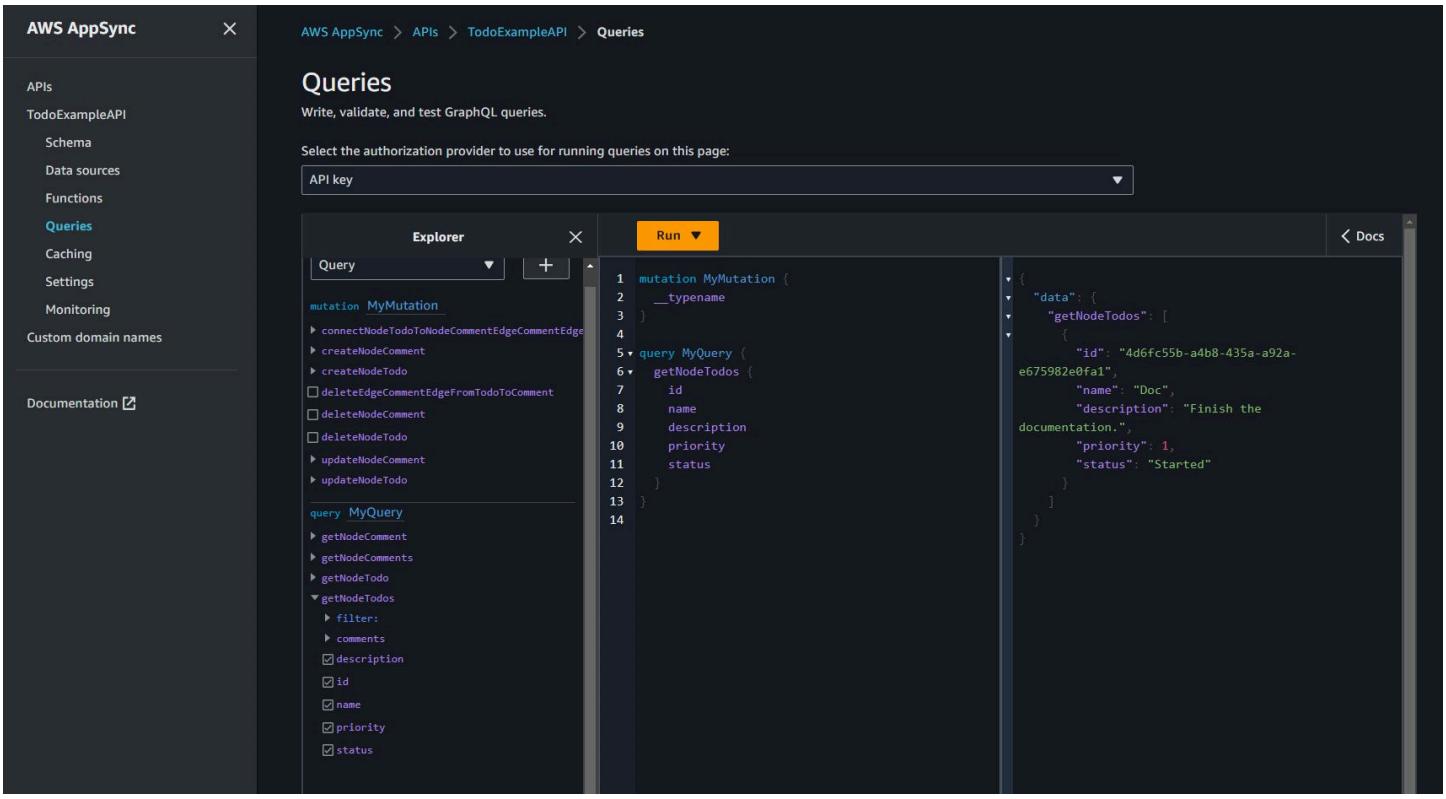
type Mutation {
  createNodeTodo(input: TodoInput!): Todo
  updateNodeTodo(input: TodoInput!): Todo
  deleteNodeTodo(_id: ID!): Boolean
  connectNodeTodoToNodeCommentEdgeCommentEdge(from_id: ID!, to_id: ID!): CommentEdge
  deleteEdgeCommentEdgeFromTodoToComment(from_id: ID!, to_id: ID!): Boolean
  createNodeComment(input: CommentInput!): Comment
  updateNodeComment(input: CommentInput!): Comment
  deleteNodeComment(_id: ID!): Boolean
}

schema {
  query: Query
  mutation: Mutation
}
```

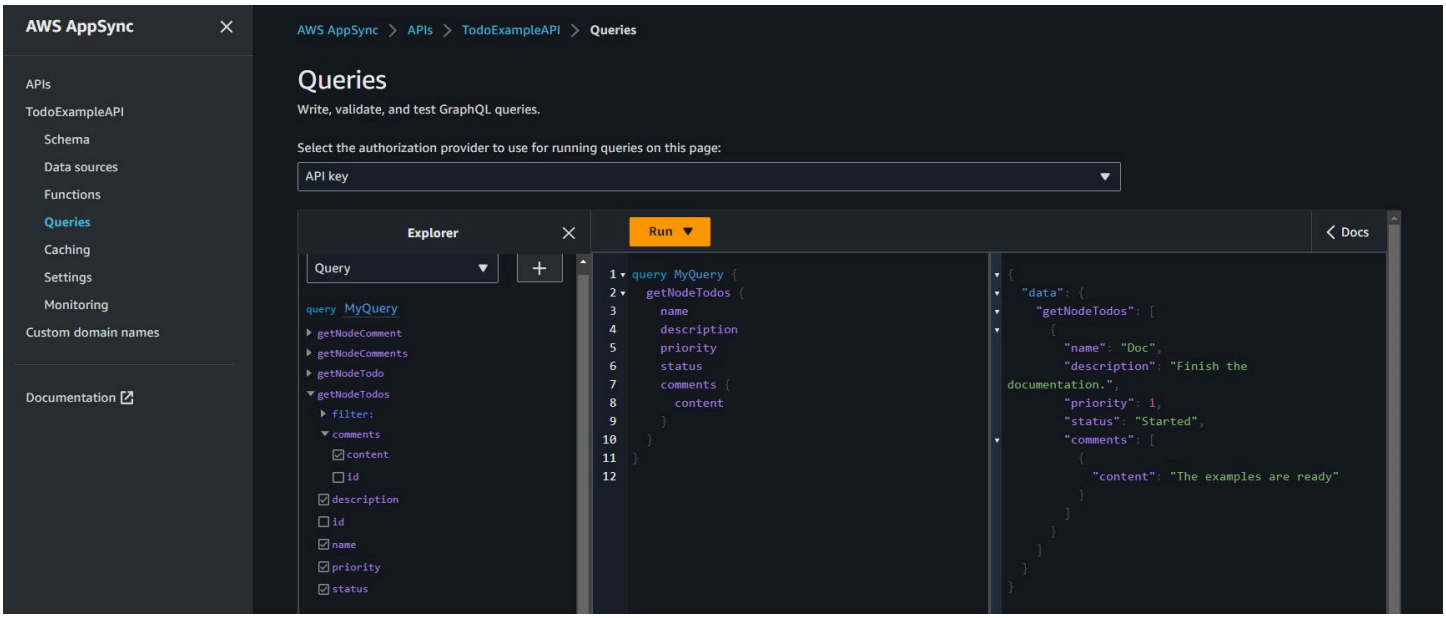
Agora você pode criar e consultar dados. Aqui está um instantâneo do console de AppSync consultas usado para testar a nova API GraphQL, chamada `TodoExampleAPI` nesse caso. Na janela central, o Explorador mostra uma lista de consultas e mutações das quais você pode escolher uma consulta, os parâmetros de entrada e os campos de retorno. Esta captura de tela mostra a criação de um tipo de nó Todo usando a mutação `createNodeTodo`:



Esta captura de tela mostra a consulta de todos os nós Todo usando a consulta `getNodeTodos`:



Depois de criar um comentário usando `createNodeComment`, você pode usar mutação `connectNodeTodoToNodeCommentEdgeCommentEdge` para conectá-los especificando seus IDs. Aqui está uma consulta aninhada para recuperar Todos e seus comentários anexados:



Se quiser fazer alterações no arquivo `TodoExample.source.graphql` conforme descrito em Trabalhar com diretivas, você poderá usar o esquema editado como entrada e executar o utilitário novamente. O utilitário então modificará a API GraphQL adequadamente.

Trabalhar com diretivas para um esquema do GraphQL

É possível partir de um esquema do GraphQL que já tenha diretivas, usando um comando como o seguinte:

```
neptune-for-graphql \
  --input-schema-file (your GraphQL schema file with directives) \
  --create-update-aws-pipeline \
  --create-update-aws-pipeline-name (name for your new GraphQL API) \
  --create-update-aws-pipeline-neptune-endpoint  (empty Neptune database
 endpoint):(port number) \
  --output-resolver-query-https
```

Você pode modificar as diretivas que o utilitário criou ou adicionar as próprias diretivas a um esquema do GraphQL. Aqui estão algumas maneiras de trabalhar com diretivas:

Executar o utilitário para que ele não gere mutações

Para evitar que o utilitário gere mutações na API do GraphQL, use a opção `--output-schema-no-mutations` no comando `neptune-for-graphql`.

A diretiva `@alias`

A diretiva `@alias` pode ser aplicada aos tipos ou aos campos do esquema do GraphQL. Ele associa nomes diferentes entre o banco de dados de grafos e o esquema do GraphQL. A sintaxe é:

```
@alias(property: (property name))
```

No exemplo abaixo, `airport` é o rótulo do nó do banco de dados de grafos associado ao tipo `Airport` do GraphQL e `desc` é a propriedade do nó gráfico associada ao campo `description` (veja o Exemplo de rotas aéreas):

```
type Airport @alias(property: "airport") {
  city: String
  description: String @alias(property: "desc")
}
```

Observe que a formatação padrão do GraphQL exige nomes de tipo Pascal-casing e nomes de campo camel-casing.

A diretiva `@relationship`

A diretiva `@relationship` associa tipos de GraphQL a bordas do banco de dados de grafos. A sintaxe é:

```
@relationship(edgeType: (edge name), direction: (IN or OUT))
```

Veja a seguir um exemplo de comando:

```
type Airport @alias(property: "airport") {
  ...
  continentContainsIn: Continent @relationship(edgeType: "contains", direction: IN)
  countryContainsIn: Country @relationship(edgeType: "contains", direction: IN)
  airportRoutesOut(filter: AirportInput, options: Options): [Airport]
 @relationship(edgeType: "route", direction: OUT)
```

```
  airportRoutesIn(filter: AirportInput, options: Options): [Airport]
 @relationship(edgeType: "route", direction: IN)
}
```

Você pode encontrar diretivas `@relationship` no Exemplo de Todo e no Exemplo de rotas aéreas.

As diretivas **`@graphQuery`** e **`@cypher`**

É possível definir consultas do openCypher para resolver um valor de campo, bem como adicionar consultas ou mutações. Por exemplo, isso adiciona um novo campo `outboundRoutesCount` ao tipo `Airport` para contar as rotas de saída:

```
type Airport @alias(property: "airport") {
  ...
  outboundRoutesCount: Int @graphQuery(statement: "MATCH (this)-[r:route]->(a) RETURN
 count(r)")
}
```

Aqui está um exemplo de novas consultas e mutações:

```
type Query {
  getAirportConnection(fromCode: String!, toCode: String!): Airport \
    @cypher(statement: \
      "MATCH (:airport{code: '$fromCode'})-[:route]->(this:airport)-[:route]-
>(:airport{code:'$toCode'})")
}

type Mutation {
  createAirport(input: AirportInput!): Airport @graphQuery(statement: "CREATE
 (this:airport {$input}) RETURN this")
  addRoute(fromAirportCode:String, toAirportCode:String, dist:Int): Route \
    @graphQuery(statement: \
     "MATCH (from:airport{code:'$fromAirportCode'}),
 (to:airport{code:'$toAirportCode'}) \
      CREATE (from)-[this:route{dist:$dist}]->(to) \
      RETURN this")
}
```

Observe que, se você omitir o `RETURN`, o resolvedor assumirá que a palavra-chave `this` é o escopo de retorno.

Você também pode adicionar uma consulta ou uma mutação usando uma consulta do Gremlin:

```
type Query {
  getAirportWithGremlin(code:String): Airport \
    @graphQuery(statement: "g.V().has('airport', 'code', '$code').elementMap()")  #
 single node
  getAirportsWithGremlin: [Airport] \
    @graphQuery(statement: "g.V().hasLabel('airport').elementMap().fold()")       #
 list of nodes
  getCountriesCount: Int \
    @graphQuery(statement: "g.V().hasLabel('country').count()")                   #
 scalar
}
```

No momento, as consultas do Gremlin são limitadas às que geram valores escalares, ou `elementMap()` para um único nó, ou `elementMap().fold()` para uma lista de nós.

A diretiva **`@id`**

A diretiva `@id` identifica o campo associada à entidade do banco de dados `id` de grafos. Bancos de dados de grafos, como o Amazon Neptune, sempre têm um `id` exclusivo para nós e bordas que é atribuído durante importações em massa ou gerado automaticamente. Por exemplo: .

```
type Airport {
  _id: ID! @id
  city: String
  code: String
}
```

Nomes reservados de tipo, consulta e mutação

O utilitário gera automaticamente consultas e mutações para criar uma API do GraphQL funcional. O padrão desses nomes é reconhecido pelo resolvedor e é reservado. Aqui estão alguns exemplos do tipo `Airport` e do tipo de conexão `Route`:

O tipo `Options` é reservado.

```
input Options {
  limit: Int
}
```

Os parâmetros de função `filter` e `options` são reservados.

```
type Query {
  getNodeAirports(filter: AirportInput, options: Options): [Airport]
}
```

O prefixo getNode dos nomes das consultas é reservado, e os prefixos de nomes de mutações, como `createNode`, `updateNode`, `deleteNode`, `connectNode`, `deleteNode`, `updateEdge` e `deleteEdge` são reservados.

```
type Query {
  getNodeAirport(id: ID, filter: AirportInput): Airport
  getNodeAirports(filter: AirportInput): [Airport]
}

type Mutation {
  createNodeAirport(input: AirportInput!): Airport
  updateNodeAirport(id: ID!, input: AirportInput!): Airport
  deleteNodeAirport(id: ID!): Boolean
  connectNodeAirportToNodeAirportEdgeRout(from: ID!, to: ID!, edge: RouteInput!): Route
  updateEdgeRouteFromAirportToAirport(from: ID!, to: ID!, edge: RouteInput!): Route
  deleteEdgeRouteFromAirportToAirport(from: ID!, to: ID!): Boolean
}
```

Aplicar alterações ao esquema do GraphQL

É possível modificar o esquema de origem do GraphQL e executar o utilitário novamente, obtendo o esquema mais recente do seu banco de dados Neptune. Toda vez que o utilitário descobre um novo esquema no banco de dados, ele gera um novo esquema do GraphQL.

Você também pode editar manualmente o esquema de origem do GraphQL e executar o utilitário novamente usando o esquema de origem como entrada em vez do endpoint do banco de dados Neptune.

Por fim, é possível colocar suas alterações em um arquivo usando este formato JSON:

```
[
  {
    "type": "(GraphQL type name)",
    "field": "(GraphQL field name)",
    "action": "(remove or add)",
    "value": "(value)"
```

```
  }
]
```

Por exemplo: .

```
[
  {
    "type": "Airport",
    "field": "outboundRoutesCountAdd",
    "action": "add",
    "value":"outboundRoutesCountAdd: Int @graphQuery(statement: \"MATCH (this)-
[r:route]->(a) RETURN count(r)\")"
  },
  {
    "type": "Mutation",
    "field": "deleteNodeVersion",
    "action": "remove",
    "value": ""
  },
  {
    "type": "Mutation",
    "field": "createNodeVersion",
    "action": "remove",
    "value": ""
  }
]
```

Depois, ao executar o utilitário nesse arquivo usando o parâmetro `--input-schema-changes-file` no comando, o utilitário aplica suas alterações de uma só vez.

Argumentos de linha de comando para o utilitário GraphQL

- **`--help, -h`**: gera o texto de ajuda do utilitário GraphQL para o console.

- **`--input-schema`** ***`(schema text)`***: um esquema do GraphQL, com ou sem diretivas, para usar como entrada.

- **`--input-schema-file`** ***`(file URL)`***: o URL de um arquivo que contém um esquema do GraphQL para usar como entrada.

- **`--input-schema-changes-file` *`(file URL)`***: o URL de um arquivo que contém as alterações que você deseja fazer em um esquema do GraphQL. Se você executar o utilitário em um banco de dados Neptune várias vezes e também alterar manualmente o esquema de origem do GraphQL, talvez adicionando uma consulta personalizada, suas alterações manuais serão perdidas. Para evitar isso, coloque suas alterações em um arquivo de alterações e passe-o usando esse argumento.

  O arquivo de alterações requer o seguinte formato JSON:

```
[
  {
    "type": "(GraphQL type name)",
    "field": "(GraphQL field name)",
    "action": "(remove or add)",
    "value": "(value)"
  }
]
```

  Para obter mais informações, consulte o Exemplo de Todo.

- **`--input-graphdb-schema` *`(schema text)`***: em vez de executar o utilitário em um banco de dados Neptune, você pode expressar um esquema graphdb em formato de texto para usar como entrada. Um esquema graphdb tem um formato JSON como este:

```
{
  "nodeStructures": [
    { "label":"nodelabel1",
      "properties": [
        { "name":"name1", "type":"type1" }
      ]
    },
    { "label":"nodelabel2",
      "properties": [
          { "name":"name2", "type":"type1" }
      ]
    }
  ],
  "edgeStructures": [
```

```
    {
      "label":"label1",
      "directions": [
        { "from":"nodelabel1", "to":"nodelabel2", "relationship":"ONE-ONE|ONE-MANY|
MANY-MANY"  }
      ],
      "properties": [
        { "name":"name1", "type":"type1" }
      ]
    }
  ]
}
```

- **`--input-graphdb-schema-file`** ***`(file URL)`***: em vez de executar o utilitário em um banco de dados Neptune, você pode expressar um esquema graphdb em um texto para usar como entrada. Consulte `--input-graphdb-schema` acima para obter um exemplo do formato JSON para um arquivo de esquema graphdb.

- **`--input-graphdb-schema-neptune-endpoint`** ***`(endpoint URL)`***: o enpoint do banco de dados Neptune do qual o utilitário deve extrair o esquema graphdb.

- **`--output-schema-file`** ***`(file name)`***: o nome do arquivo de saída para o esquema GraphQL. Se não for especificado, o padrão será `output.schema.graphql`, a menos que um nome de pipeline tenha sido definido usando `--create-update-aws-pipeline-name`. Nesse caso, o nome do arquivo padrão será *`(pipline name)`*`.schema.graphql`.

- **`--output-source-schema-file`** ***`(file name)`***: o nome do arquivo de saída para o esquema do GraphQL com diretivas. Se não for especificado, o padrão será `output.source.schema.graphql`, a menos que um nome de pipeline tenha sido definido usando `--create-update-aws-pipeline-name`. Nesse caso, o nome padrão será *`(pipeline name)`*`.source.schema.graphql`.

- **`--output-schema-no-mutations`**: se esse argumento estiver presente, o utilitário não vai gerar mutações na API do GraphQL, apenas consultas.

- **`--output-neptune-schema-file` *`(file name)`***: o nome do arquivo de saída do esquema graphdb do Neptune descoberto pelo utilitário. Se não for especificado, o padrão será `output.graphdb.json`, a menos que um nome de pipeline tenha sido definido usando `--create-update-aws-pipeline-name`. Nesse caso, o nome do arquivo padrão será *`(pipeline name)`*`.graphdb.json`.

- **`--output-js-resolver-file` *`(file name)`***: o nome do arquivo de saída para uma cópia do código do resolvedor. Se não for especificado, o padrão será `output.resolver.graphql.js`, a menos que um nome de pipeline tenha sido definido usando `--create-update-aws-pipeline-name`. Nesse caso, o nome do arquivo será *`(pipeline name)`*`.resolver.graphql.js`.

  Esse arquivo é compactado no pacote de código carregado na função do Lambda que executa o resolvedor.

- **`--output-resolver-query-sdk`**: esse argumento especifica que a função do Lambda do utilitário deve consultar o Neptune usando o SDK de dados do Neptune, que está disponível a partir da versão 1.2.1.0.R5 do mecanismo do Neptune (esse é o padrão). No entanto, se o utilitário detectar uma versão mais antiga do mecanismo do Neptune, ele vai sugerir o uso da opção HTTPS do Lambda, que você pode invocar usando o argumento `--output-resolver-query-https`.

- **`--output-resolver-query-https`**: esse argumento especifica que a função do Lambda do utilitário deve consultar o Neptune usando a API HTTPS do Neptune.

- **`--create-update-aws-pipeline`**— Esse argumento aciona a criação dos AWS recursos para uso da API GraphQL, incluindo a API GraphQL e o Lambda que AppSync executa o resolvedor.

- **`--create-update-aws-pipeline-name` *`(pipeline name)`***— Esse argumento define o nome do pipeline, como a `pipeline-name` API AppSync ou a `pipeline-name` função da função

Lambda. Se um nome não for especificado, `--create-update-aws-pipeline` usará o nome do banco de dados `Neptune` .

- **`--create-update-aws-pipeline-region` *`(AWS region)`***: esse argumento define a região AWS na qual o pipeline da API do GraphQL é criado. Se não for especificada, a região padrão será `us-east-1` ou a região em que o banco de dados Neptune está localizado, extraída do endpoint do banco de dados.
- **`--create-update-aws-pipeline-neptune-endpoint` *`(endpoint URL)`***: esse argumento define o endpoint do banco de dados Neptune usado pela função do Lambda para consultar o banco de dados. Se não for definido, o endpoint definido por `--input-graphdb-schema-neptune-endpoint` será usado.
- **`--remove-aws-pipeline-name` *`(pipeline name)`***: esse argumento remove um pipeline criado usando `--create-update-aws-pipeline`. Os recursos a serem removidos estão listados em um arquivo chamado *`(pipeline name)`*`.resources.json`.
- **`--output-aws-pipeline-cdk`**— Esse argumento aciona a criação de um arquivo CDK que pode ser usado para criar os AWS recursos para a API GraphQL, incluindo a AppSync API GraphQL e a função Lambda que executa o resolvedor.
- **`--output-aws-pipeline-cdk-neptume-endpoint` *`(endpoint URL)`***: esse argumento define o endpoint do banco de dados Neptune usado pela função do Lambda para consultar o banco de dados Neptune. Se não for definido, o endpoint definido por `--input-graphdb-schema-neptune-endpoint` será usado.
- **`--output-aws-pipeline-cdk-name` *`(pipeline name)`***— Esse argumento define o nome do pipeline para a AppSync API e a função Lambda pipeline-name a serem usados. Se não for especificado, `--create-update-aws-pipeline` usará o nome do banco de dados Neptune.
- **`--output-aws-pipeline-cdk-region` *`(AWS region)`***: isso define a região AWS na qual o pipeline da API do GraphQL é criado. Se não for especificada, o padrão será `us-east-1` ou a região em que o banco de dados Neptune está localizado, extraída do endpoint do banco de dados.
- **`--output-aws-pipeline-cdk-file` *`(file name)`***: isso define o nome do arquivo CDK. Se não estiver definido, o padrão será *`(pipeline name)`*`-cdk.js`.

Erros do serviço do Neptune

O Amazon Neptune tem dois conjuntos diferentes de erros:

- Os erros do mecanismo de grafo que são relativos apenas aos endpoints do cluster de banco de dados do Neptune.
- Os erros que são associados às APIs para criar e modificar recursos do Neptune com o AWS SDK e a AWS Command Line Interface (AWS CLI).

Tópicos

Mensagens de erro e códigos do mecanismo gráfico

Os endpoints do Amazon Neptune geram os erros padrão para Gremlin e SPARQL quando encontrados.

Os erros específicos do Neptune também podem ser gerados pelos mesmos endpoints. Esta seção documenta as mensagens de erro, os códigos e as ações recomendadas do Neptune.

> **Note**
>
> Esses erros são relativos somente aos endpoints do cluster de banco de dados do Neptune. As APIs para criar e modificar recursos do Neptune com o AWS SDK e a AWS CLI têm um conjunto diferente de erros comuns. Para obter mais informações sobre esses erros, consulte the section called “Erros de API”.

Formato de erro do mecanismo de gráfico

As mensagens de erro do Neptune geram um código de erro HTTP relevante e uma resposta em formato JSON.

```
HTTP/1.1 400 Bad Request
```

```
x-amzn-RequestId: LDM6CJP8RMQ1FHKSC1RBVJFPNVV4KQNSO5AEMF66Q9ASUAAJG
Content-Type: application/x-amz-json-1.0
Content-Length: 465
Date: Thu, 15 Mar 2017 23:56:23 GMT

{
    "requestId": "0dbcded3-a9a1-4a25-b419-828c46342e47",
    "code": "ReadOnlyViolationException",
    "detailedMessage": "The request is rejected because it violates some read-only
 restriction, such as a designation of a replica as read-only."
}
```

Erros de consulta do mecanismo de gráfico

A tabela a seguir contém o código de erro, a mensagem e o status HTTP.

Ela também indica se é aceitável repetir a solicitação. No geral, é aceitável repetir a solicitação se ela puder ser bem-sucedida em uma nova tentativa.

Neptune Service Error Code	HTTP status	Ok to Retry?	Message
`AccessDeniedException`	403	No	Authentication or authorization failure.
`BadRequestException`	400	No	The request could not be completed.
`BadRequestException`	400	No	Request size exceeds max allowed value of 157286400 bytes.
`CancelledByUserException`	500	Yes	The request processing was cancelled by an authorized client.
`ConcurrentModificationException`	500	Yes	The request processing did not succeed due to a modification conflict. The

Neptune Service Error Code	HTTP status	Ok to Retry?	Message
			client should retry the request.
`ConstraintViolationException`	400	Yes	The query engine discovered, during the execution of the request, that the completion of some operation is impossible without violating some data integrity constraints, such as persistence of in- and out-vertices while adding an edge. Such conditions are typically observed if there are concurrent modifications to the graph, and are transient. The client should retry the request.
`FailureByQueryException`	500	Yes	Calling fail() caused request processing to fail.
`InternalFailureException`	500	Yes	The request processing has failed.

Neptune Service Error Code	HTTP status	Ok to Retry?	Message
`InvalidNumericDataException`	400	No	Invalid use of numeric data which cannot be represented in 64-bit storage size.
`InvalidParameterException`	400	No	An invalid or out-of-range value was supplied for some input parameter or invalid syntax in a supplied RDF file.
`MalformedQueryException`	400	No	The request is rejected because it contains a query that is syntactically incorrect or does not pass additional validation.
`MemoryLimitExceededException`	500	Yes	The request processing did not succeed due to lack of memory, but can be retried when the server is less busy.
`MethodNotAllowedException`	405	No	The request is rejected because the chosen HTTP method is not supported by the used endpoint.

Neptune Service Error Code	HTTP status	Ok to Retry?	Message
`MissingParameterException`	400	No	A required parameter for the specified action is not supplied.
`QueryLimitExceededException`	500	Yes	The request processing did not succeed due to the lack of a limited resource, but can be retried when the server is less busy.
`QueryLimitException`	400	No	Size of query exceeds system limit.
`QueryTooLargeException`	400	No	The request was rejected because its body is too large.
`ReadOnlyViolationException`	400	No	The request is rejected because it violates some read-only restriction, such as a designation of a replica as read-only.
`ThrottlingException`	500	Yes	Rate of requests exceeds the maximum throughput. OK to retry.
`TimeLimitExceededException`	500	Yes	The request processing timed out.

Neptune Service Error Code	HTTP status	Ok to Retry?	Message
`TooManyRequestsException`	429	Yes	The rate of requests exceeds the maximum throughput. OK to retry.
`UnsupportedOperationException`	400	No	The request uses a currently unsupported feature or construct.

Erros de autenticação do IAM

Esses erros são específicos de cluster que possui autenticação do IAM habilitada.

A tabela a seguir contém o código de erro, a mensagem e o status HTTP.

Neptune Service Error Code	HTTP status	Message
Incorrect IAM User/Policy	403	You do not have sufficient access to perform this action.
Incorrect or Missing Region	403	Credential should be scoped to a valid Region, not '*região*'.
Incorrect or Missing Service Name	403	Credential should be scoped to correct service: '`neptune-db `'.
Incorrect or Missing Host Header / Invalid Signature	403	The request signature we calculated does not match the signature you provided. Check your AWS Secret Access Key and signing method. Consult the service documentation for

Neptune Service Error Code	HTTP status	Message
		details. Host header is missing or hostname is incorrect.
Missing `X-Amz-Security-Token`	403	'`x-amz-security-token`' is named as a `SignedHeader`, but it does not exist in the HTTP request
Missing Authorization Header	403	The request did not include the required authorization header, or it was malformed.
Missing Authentication Token	403	Missing Authentication Token.
Old Date	403	Signature expired: *20181011T213907Z* is now earlier than *20181011T213915Z* (*20181011T214415Z* - *Cinco min.*.)
Future Date	403	Signature not yet current: *20500224T213559Z* is still later than *20181108T225925Z* (*20181108T225425Z* + *Cinco min.*.)
Incorrect Date Format	403	Date must be in ISO-8601 'basic format'. Got '*data*'. See https://en.wikipedia.org/wiki/ISO_8601.
Unknown/Missing Access Key or Session Token	403	The security token included in the request is invalid.

Neptune Service Error Code	HTTP status	Message
Unknown/Missing Secret Key	403	The request signature we calculated does not match the signature you provided. Check your AWS Secret Access Key and signing method. Consult the service documentation for details. Host header is missing or hostname is incorrect.
`TooManyRequestsException`	429	The rate of requests exceeds the maximum throughput. OK to retry.

Códigos e mensagens de erro da API de gerenciamento do cluster de banco de dados

Esses erros do Amazon Neptune são associados às APIs para criar e modificar recursos do Neptune com o AWS SDK e a AWS CLI.

A tabela a seguir contém o código de erro, a mensagem e o status HTTP.

Neptune Service Error Code	HTTP status	Message
`AccessDeniedException`	403	Você não tem acesso suficiente para executar essa ação.
`IncompleteSignature`	400	A assinatura da solicitação não segue os padrões da AWS.
`InternalFailure`	500	O processamento da solicitação falhou por causa de um erro, uma exceção ou uma falha desconhecida.

Neptune Service Error Code	HTTP status	Message
`InvalidAction`	400	A ação ou operação solicitada é inválida. Verifique se a ação foi digitada corretamente.
`InvalidClientTokenId`	403	O certificado X.509 ou o ID de chave de acesso da AWS fornecido não existe em nossos registros.
`InvalidParameterCombination`	400	Parâmetros que não devem ser usados em conjunto foram usados em conjunto.
`InvalidParameterValue`	400	Um valor inválido ou fora do intervalo foi fornecido para o parâmetro de entrada.
`InvalidQueryParameter`	400	Um valor inválido ou fora do intervalo foi fornecido para o parâmetro de entrada.
`MalformedQueryString`	400	A string de consulta contém um erro de sintaxe.
`MissingAction`	400	A solicitação carece de uma ação ou de um parâmetro necessário.
`MissingAuthenticationToken`	403	A solicitação deve conter um ID de chave de acesso da AWS válido (registrado) ou um certificado X.509.
`MissingParameter`	400	Um parâmetro obrigatório para a ação especificada não foi fornecido.

Neptune Service Error Code	HTTP status	Message
`OptInRequired`	403	O ID da chave de acesso da AWS precisa de uma assinatura do serviço.
`RequestExpired`	400	A solicitação atingiu o serviço mais de 15 minutos após a data na solicitação ou mais de 15 minutos após a data de expiração da solicitação (como para pre-signed URLs), ou a data na solicitação está a mais de 15 minutos no futuro.
`ServiceUnavailable`	503	Falha na solicitação devido a um erro temporário do servidor.
`ThrottlingException`	500	A solicitação foi negada devido à limitação da solicitação.
`ValidationError`	400	A entrada não atende às restrições especificadas por um serviço da AWS.

Erro de carregador do Neptune e mensagens de feed

As seguintes mensagens são retornados pelo endpoint `status` do carregador do Neptune. Para obter mais informações, consulte API Get-Status.

A tabela a seguir contém o código e a descrição do feed do carregador.

Error or Feed Code	Description
LOAD_NOT_STARTED	A carga foi registrada, mas não foi iniciada.

Error or Feed Code	Description
LOAD_IN_PROGRESS	Indica que o carregamento está em andamento e especifica o número de arquivos que estão sendo carregados no momento. Quando o carregador analisa um arquivo, ele cria um ou mais blocos a serem carregados em paralelo. Como um único arquivo pode produzir vários fragmentos, a contagem de arquivos incluída nessa mensagem geralmente é menor que o número de threads usados pelo processo de carregamento em massa.
LOAD_COMPLETED	A carga foi concluída sem erros ou os erros estão dentro de um limite aceitável.
LOAD_CANCELLED_BY_USER	A carga foi cancelada pelo usuário.
LOAD_CANCELLED_DUE_TO_ERRORS	A carga foi cancelada pelo sistema devido a erros.
LOAD_UNEXPECTED_ERROR	Houve falha na carga com um erro inesperado.
LOAD_FAILED	O carregamento falhou como resultado de um ou mais erros.
LOAD_S3_READ_ERROR	Falha no feed devido a problemas de conectividade do Amazon S3; intermitentes ou transitórios. Se qualquer um dos feeds receberem esse erro, o status geral da carga será definido como LOAD_FAILED.
LOAD_S3_ACCESS_DENIED_ERROR	O acesso ao bucket do S3 foi negado. Se qualquer um dos feeds receberem esse erro, o status geral da carga será definido como LOAD_FAILED.

Error or Feed Code	Description
LOAD_COMMITTED_W_WRITE_CONFLICTS	Dados carregados confirmados com conflitos de gravação não resolvidos. O carregador tentará resolver os conflitos de gravação em transações separadas e atualizar á o status do feed conforme a carga progredir . Se o status final do feed for LOAD_COMM ITTED_W_WRITE_CONFLICTS, tente retomar a carga e ela provavelmente terá êxito sem conflitos de gravação. Um conflito de gravação não está normalmente relacionado a dados de entrada inválidos, mas as duplicações nos dados podem aumentar a probabilidade de conflitos de gravação.
LOAD_DATA_DEADLOCK	Load was automatically rolled back due to deadlock.
LOAD_DATA_FAILED_DUE_TO_FEE D_MODIFIED_OR_DELETED	O feed falhou porque o arquivo foi excluído ou atualizado após o início do carregamento.
LOAD_FAILED_BECAUSE_DEPENDE NCY_NOT_SATISFIED	A solicitação de carga não foi executada devido à falha da verificação.
LOAD_IN_QUEUE	A solicitação de carga foi enfileira e está aguardando execução.
LOAD_FAILED_INVALID_REQUEST	A carga falhou porque a solicitação era inválida (por exemplo, a fonte/o bucket especificado pode não existir ou o formato do arquivo é inválido).

Versões do mecanismo do Amazon Neptune

O Amazon Neptune lança atualizações do mecanismo regularmente.

É possível determinar qual versão do mecanismo está instalada no momento usando a API de status da instância ou o console do Neptune. O número da versão informa se você está executando uma versão principal original, uma versão secundária ou uma versão de patch. Para obter mais informações sobre a numeração de versões, consulte Números das versões do mecanismo.

Para obter mais informações sobre atualizações em geral, consulte Manutenção do cluster.

A partir da versão 1.3.0.0 do mecanismo, as versões terão a estrutura mostrada na tabela abaixo. O número da versão secundária é o que será avaliado para o processamento `AutoMinorVersionUpgrade`.

Version (Versão)	Versã do produ	Versã princ	Versão secundári a	Versão do patch	Status	Liberado	Fim da vida útil	Atualizar para:
1.3.2.1	1	3	2	1	ativo	2024-06-20	2025-11-30	N/D
1.3.2.0	1	3	2	0	ativo	2024-06-10	2025-11-30	1.3.2.1
1.3.1.0	1	3	1	0	ativo	2024-03-06	2025-11-30	1.3.2.1
1.3.0.0	1	3	0	0	ativo	2023-11-15	2025-11-30	1.3.2.1

A tabela abaixo lista todas as versões do mecanismo desde 1.0.1.0, junto com informações sobre a versão. end-of-life É possível usar as datas desta tabela para planejar os ciclos de teste e atualização.

Version (Versão)	Versão principa	Versão secunda	Status	Liberado	Fim da vida útil	Atualizar para:
1.2.1.1	1.2	1.1	ativo	2024-03-11	2025-03-06	1.3.0.0
1.2.1.0	1.2	1,0	ativo	2023-03-08	2025-03-06	1.3.0.0
1.2.0.2	1.2	0.2	ativo	2022-11-16	2025-03-06	1.3.0.0
1.2.0.1	1.2	0.1	ativo	2022-10-26	2025-03-06	1.3.0.0
1.2.0.0	1.2	0.0	ativo	2022-07-21	2025-03-06	1.3.0.0
1.1.1.0	1.1	1,0	ativo	2022-04-19	31/10/2024	1.2.1.0
1.1.0.0	1.1	0.0	ativo	2021-11-19	31/10/2024	1.1.1.0
1.0.5.1	1,0	5.1	obsoleto	2021-10-01	2023-01-30	1.1.0.0
1.0.5.0	1,0	5,0	obsoleto	2021-07-27	2023-01-30	1.1.0.0
1.0.4.2	1,0	4.2	obsoleto	2021-06-01	2023-01-30	1.1.0.0
1.0.4.1	1,0	4.1	obsoleto	2020-12-08	2023-01-30	1.1.0.0
1.0.4.0	1,0	4,0	obsoleto	2020-10-12	2023-01-30	1.1.0.0
1.0.3.0	1,0	3.0	obsoleto	2020-08-03	2023-01-30	1.1.0.0
1.0.2.2	1,0	2.2	obsoleto	2020-03-09	2022-07-29	1.0.3.0
1.0.2.1	1,0	2.1	obsoleto	22/11/2019	2022-07-29	1.0.3.0
1.0.2.0	1,0	2,0	obsoleto	08/11/2019	2020-05-19	1.0.3.0
1.0.1.2	1,0	1.2	obsoleto	2019-10-15	—	—
1.0.1.1	1,0	1.1	obsoleto	13/08/2019	—	—

Version (Versão)	Versão principa	Versão secunda	Status	Liberado	Fim da vida útil	Atualizar para:
1.0.1.0.*	1,0	1.0.*	obsoleto	02/07/2019 e antes	—	—

end-of-life Planejamento da versão principal do motor

As versões do mecanismo do Neptune quase sempre chegam ao fim da vida útil no final de um trimestre civil. As exceções ocorrem somente quando surgem problemas importantes de segurança ou disponibilidade.

Quando uma versão do mecanismo chegar ao fim da vida útil, você precisará atualizar o banco de dados Neptune para uma versão mais recente.

Em geral, as versões do mecanismo do Neptune continuam disponíveis da seguinte forma:

- Versões secundárias do mecanismo: as versões secundárias do mecanismo permanecem disponíveis por pelo menos seis meses após o lançamento.
- Versões principais do mecanismo: as versões principais do mecanismo permanecem disponíveis por pelo menos 12 meses após o lançamento.

Pelo menos 3 meses antes de uma versão do motor chegar ao fim de sua vida útil, AWS enviará uma notificação automática por e-mail para o endereço de e-mail associado à sua AWS conta e publicará a mesma mensagem no seu AWS Health Dashboard. Isso dará tempo para planejar e se preparar para a atualização.

Quando uma versão do mecanismo chegar ao fim da vida útil, você não poderá mais criar clusters ou instâncias usando essa versão, nem o ajuste de escala automático poderá criar instâncias usando essa versão.

Uma versão do mecanismo que realmente chegue ao fim da vida útil será atualizada automaticamente durante uma janela de manutenção. A mensagem enviada a você três meses antes do fim da vida útil da versão do mecanismo conterá detalhes sobre o que essa atualização automática envolverá, incluindo a versão para a qual o mecanismo será atualizado automaticamente, o impacto nos clusters de banco de dados e as ações que recomendamos.

Important

Você é responsável por manter as versões do mecanismo de banco de dados atualizadas. A AWS incentiva todos os clientes a atualizar os bancos de dados para a versão mais recente do mecanismo, a fim de se beneficiarem das proteções de segurança, privacidade e disponibilidade mais atualizadas. Se você operar o banco de dados em um mecanismo ou um software não compatível após a data de defasagem ("Mecanismo herdado"), enfrentará uma maior probabilidade de riscos operacionais, segurança e privacidade, incluindo eventos de inatividade.

A operação do seu banco de dados em qualquer mecanismo está sujeita ao Contrato que rege o uso dos AWS Serviços. Os motores antigos não estão disponíveis ao público em geral. AWS não fornece mais suporte para o Legacy Engine e AWS pode limitar o acesso ou o uso de qualquer Legacy Engine a qualquer momento, se AWS determinar que o Legacy Engine representa um risco de segurança ou responsabilidade, ou um risco de danos, aos Serviços AWS, a suas Afiliadas ou a terceiros. Sua decisão de continuar executando Seu conteúdo em um Mecanismo herdado pode fazer com que Seu conteúdo fique indisponível, corrompido ou irrecuperável. Os bancos de dados executados em um Mecanismo herdado estão sujeitos às exceções do Acordo de Serviço (SLA).

BANCOS DE DADOS E SOFTWARES RELACIONADOS EXECUTADOS EM UM MECANISMO HERDADO CONTÊM BUGS, ERROS, DEFEITOS E/OU COMPONENTES NOCIVOS. CONSEQUENTEMENTE, E NÃO OBSTANTE QUALQUER DISPOSIÇÃO EM CONTRÁRIO NO CONTRATO OU NOS TERMOS DO SERVIÇO, AWS ESTÁ FORNECENDO O MECANISMO ANTIGO "NO ESTADO EM QUE SE ENCONTRA".

Amazon Neptune Engine versão 1.3.2.1 (2024-06-20)

Em 2024-06-20, a versão 1.3.2.1 do mecanismo geralmente está sendo implantada. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Note

A versão 1.3.0.0 do mecanismo introduziu um novo formato para grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados. Como resultado, se você estiver atualizando de uma versão de mecanismo anterior à 1.3.0.0 para a versão 1.3.0.0 ou posterior, deverá recriar todos os grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados existentes usando a família de grupos de parâmetros

`neptune1.3`. As versões anteriores usavam a família de grupos de parâmetros `neptune1` ou `neptune1.2`, e esses grupos de parâmetros não funcionarão com a versão 1.3.0.0 e posterior. Consulte Grupos de parâmetros do Amazon Neptune Para mais informações.

Defeitos corrigidos nesta versão do motor

Correções do openCypher

- Foi detectado um bug no recurso de cache do plano de consulta para consultas parametrizadas que contêm uma `WITH` cláusula interna com `SKIP` e como parâmetros. `LIMIT` Os valores SKIP/LIMIT não foram parametrizados adequadamente e, como resultado, as execuções subsequentes do mesmo plano de consulta em cache com valores de parâmetros diferentes ainda retornariam os mesmos resultados da primeira execução. Isso foi corrigido.

```
# insert some nodes
UNWIND range(1, 10) as i CREATE (s {name: i}) RETURN s

# sample query
MATCH (p)
WITH p ORDER BY p.name SKIP $s LIMIT $l
RETURN p.name as res

# first time executing with {"s": 2, "l": 1}
{
  "results" : [ {
    "res" : 3
  } ]
}

# second time executing with {"s": 2, "l": 10}
# due to bug, produces
{
  "results" : [ {
    "res" : 3
  } ]
}
# with fix, produces correct results:
{
  "results" : [ {
    "res" : 3
```

```
  }, {
    "res" : 4
  }, {
    "res" : 5
  }, {
    "res" : 6
  }, {
    "res" : 7
  }, {
    "res" : 8
  }, {
    "res" : 9
  }, {
    "res" : 10
  } ]
}%
```

- Corrigido um bug em que consultas de mutação parametrizadas lançavam um `InternalFailureException` quando o parâmetro que foi passado ainda não estava presente no banco de dados.
- Corrigido um bug em que as consultas parametrizadas do Bolt ficavam travadas após atingirem uma condição de corrida durante a limpeza dos recursos da consulta.

Alterações na versão 1.3.2.1 transferidas da versão 1.3.2.0

Melhorias trazidas da versão 1.3.2.0 do motor

Melhorias gerais

- Support para TLS versão 1.3, incluindo pacotes de criptografia TLS_AES_128_GCM_SHA256 e TLS_AES_256_GCM_SHA384. O TLS 1.3 é uma opção - o TLS 1.2 ainda é o mínimo.
- O suporte estendido do OpenCypher para o formato dateime está em lab_mode para esta versão. Nós encorajamos você a testá-lo.

Melhorias no Gremlin

- TinkerPop Atualização 3.7.x
  - Fornece uma grande expansão da linguagem Gremlin.
    - Novas etapas para processar cadeias de caracteres, listas e datas.

    - Nova sintaxe para especificar a cardinalidade na etapa. `mergeV()`
    - `union()`agora pode ser usado como uma etapa inicial.
    - Para saber mais sobre as mudanças na versão 3.7.x, consulte a documentação de TinkerPop atualização.
  - Ao atualizar os drivers da linguagem Gremlin do cliente para Java, observe que as classes do serializador passaram por algumas renomeações. Você precisará atualizar a nomenclatura do pacote e da classe em seus arquivos de configuração e no código, se especificado.
- `StrictTimeoutValidation`(somente quando ativado `StrictTimeoutValidation` por meio do labmode, incluindo`StrictTimeoutValidation=enabled`): quando o `StrictTimeoutValidation` parâmetro tem um valor de`enabled`, um valor de tempo limite por consulta especificado como uma opção de solicitação ou uma dica de consulta não pode exceder o valor definido globalmente no grupo de parâmetros. Nesse caso, Netuno lançará um. `InvalidParameterException` Essa configuração pode ser confirmada em uma resposta no `/status` endpoint quando o valor é`disabled`, e nas versões 1.3.2.0 e 1.3.2.1 do Neptune, o valor padrão desse parâmetro é. `Disabled`

Melhorias no openCypher

- Consultas de baixa latência e melhoria no desempenho da taxa de transferência: melhorias gerais no desempenho para consultas OpenCypher de baixa latência. A nova versão também melhora a taxa de transferência dessas consultas. As melhorias são mais significativas quando consultas parametrizadas são usadas.
- Support for Query Plan Cache: Quando uma consulta é enviada ao Neptune, a string de consulta é analisada, otimizada e transformada em um plano de consulta, que é executado pelo mecanismo. Os aplicativos geralmente são apoiados por padrões de consulta comuns que são instanciados com valores diferentes. O cache do plano de consulta pode reduzir a latência geral ao armazenar em cache os planos de consulta e, assim, evitar a análise e a otimização desses padrões repetidos.
- Melhoria de desempenho para consultas de agregação DISTINCT.
- Melhoria de desempenho para junções envolvendo variáveis anuláveis.
- Melhoria de desempenho para consultas envolvendo valores diferentes do predicado id (nó/relacionamento).
- Suporte estendido para a funcionalidade de data e hora (ativado apenas no modo `DatetimeMillisecond` de laboratório, incluindo`DatetimeMillisecond=enabled`. Para

ter mais informações, consulte Suporte temporal na implementação do Neptune OpenCypher (Neptune Analytics e Neptune Database 1.3.2.0 e versões posteriores).

Correções de defeitos transferidas da versão 1.3.2.0 do motor

Melhorias gerais

- A mensagem de erro do NeptuneML foi atualizada ao validar o acesso aos buckets do Graphlytics.

Correções do Gremlin

- Foram corrigidas as informações de rótulos ausentes na tradução de consultas do DFE, para cenários em que etapas que não contribuem com caminhos contêm rótulos. Por exemplo: .

```
g.withSideEffect('Neptune#useDFE', true).
  V().
  has('name', 'marko').
  has("name", TextP.regex("mark.*")).as("p1").
  not(out().has("name", P.within("peter"))).
  out().as('p2').
  dedup('p1', 'p2')
```

- Corrigido um `NullPointerException` erro na tradução da consulta do DFE, que ocorre quando uma consulta é executada em dois fragmentos do DFE e o primeiro fragmento é otimizado para um nó insatisfatório. Por exemplo: .

```
g.withSideEffect('Neptune#useDFE', true).
  V().
  has('name', 'doesNotExists').
  has("name", TextP.regex("mark.*")).
  inject(1).
  V().
  out().
  has('name', 'vadas')
```

- Corrigido um bug em que Neptune podia lançar `InternalFailureException` um quando uma consulta ValueTraversal continha um modulador by () e sua entrada era Map. Por exemplo: .

```
g.V().
  hasLabel("person").
```

```
  project("age", "name").by("age").by("name").
  order().by("age")
```

Correções do openCypher

- Operações UNWIND aprimoradas (por exemplo, expandir uma lista de valores em valores individuais) para ajudar a evitar situações de falta de memória (OOM). Por exemplo: .

```
MATCH (n)-->(m)
WITH collect(m) AS list
UNWIND list AS m
RETURN m, list
```

- Otimização de identificação personalizada corrigida no caso de várias operações MERGE em que a identificação é injetada via UNWIND. Por exemplo: .

```
UNWIND [{nid: 'nid1', mid: 'mid1'}, {nid: 'nid2', mid: 'mid2'}] as ids
MERGE (n:N {`~id`: ids.nid})
MERGE (m:M {`~id`: ids.mid})
```

- Corrigiu a explosão de memória ao planejar consultas complexas com acesso à propriedade e vários saltos com relacionamentos bidirecionais. Por exemplo: .

```
MATCH (person1:person)-[:likes]->(res)-[:partOf]->(group)-[:knows]-(:entity {name:
 'foo'}),
      (person1)-[:knows]->(person2)-[:likes]-(res2), (comment)-[:presentIn]->(:Group
 {name: 'barGroup'}),
      (person1)-[:commented]->(comment2:comment)-[:partOf]->(post:Post), (comment2)-
[:presentIn]->(:Group {name: 'fooGroup'}),
      (comment)-[:contains]->(info:Details)-[:CommentType]->(:CommentType {name:
 'Positive'}),
      (comment2)-[:contains]->(info2:Details)-[:CommentType]->(:CommentType {name:
 'Positive'})
WHERE datetime('2020-01-01T00:00') <= person1.addedAfter <=
 datetime('2023-01-01T23:59') AND comment.approvedBy = comment2.approvedBy
MATCH (comment)-[:contains]->(info3:Details)-[:CommentType]->(:CommentType {name:
 'Neutral'})
RETURN person1, group.name, info1.value,  post.ranking, info3.value
```

- Consultas de agregação fixa com null como grupo por variáveis. Por exemplo: .

```
MATCH (n)
RETURN null AS group, sum(n.num) AS result
```

Correções do SPARQL

- O analisador SPARQL foi corrigido para melhorar o tempo de análise de consultas grandes, como INSERT DATA, contendo muitos triplos e tokens grandes.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.3.2.1, certifique-se de que seu projeto seja compatível com essas versões da linguagem de consulta:

- Versão compatível mais antiga do Gremlin: `3.6.2`
- Versão compatível mais recente do Gremlin: `3.7.1`
- openCypher versão: `Neptune-9.0.20190305-1.0`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.3.2.1 do motor

Você pode atualizar para esta versão a partir da versão 1.2.0.0 ou superior.

Atualizar para esta versão

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.3.2.1 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.3.2.1 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Em vez de `--apply-immediately`, é possível especificar `--no-apply-immediately`. Para realizar uma atualização de versão principal, o allow-major-version-upgrade parâmetro é necessário. Além disso, não se esqueça de incluir a versão do mecanismo ou ele poderá ser atualizado para outra versão.

Se o cluster usar um grupo de parâmetros de cluster personalizado, não se esqueça de incluir este parâmetro para especificá-lo:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Da mesma forma, se alguma instância no cluster usar um grupo de parâmetros de banco de dados personalizado, não se esqueça de incluir este parâmetro para especificá-lo:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Se você tiver alguma dúvida ou preocupação, a equipe de AWS Suporte está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Amazon Neptune Engine versão 1.3.2.0 (2024-06-10)

Em 2024-06-10, a versão 1.3.2.0 do mecanismo geralmente está sendo implantada. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

> **Note**
>
> A versão 1.3.0.0 do mecanismo introduziu um novo formato para grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados. Como resultado, se você estiver atualizando de uma versão de mecanismo anterior à 1.3.0.0 para a versão 1.3.0.0 ou posterior, deverá recriar todos os grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados existentes usando a família de grupos de parâmetros `neptune1.3`. As versões anteriores usavam a família de grupos de parâmetros `neptune1` ou `neptune1.2`, e esses grupos de parâmetros não funcionarão com a versão 1.3.0.0 e posterior. Consulte Grupos de parâmetros do Amazon Neptune Para mais informações.

> **Warning**
>
> Detectamos um problema no cache do plano de consulta quando `skip` ou `limit` é usado em uma `WITH` cláusula interna e estamos parametrizados. Para evitar esse problema, adicione a dica de consulta `QUERY:PLANCACHE "disabled"` ao enviar uma consulta que inclua uma subcláusula parametrizada de salto e/ou limite. Como alternativa, você pode codificar os valores na consulta. Para obter mais informações, consulte Atenuação do problema de cache do plano de consulta.

Melhorias nesta versão do motor

Melhorias gerais

- Support para TLS versão 1.3, incluindo pacotes de criptografia TLS_AES_128_GCM_SHA256 e TLS_AES_256_GCM_SHA384. O TLS 1.3 é uma opção - o TLS 1.2 ainda é o mínimo.

Melhorias no Gremlin

- TinkerPop Atualização 3.7.x

  - Fornece uma grande expansão da linguagem Gremlin.
    - Novas etapas para processar cadeias de caracteres, listas e datas.
    - Nova sintaxe para especificar a cardinalidade na etapa. `mergeV()`
    - `union()`agora pode ser usado como uma etapa inicial.
    - Para saber mais sobre as mudanças na versão 3.7.x, consulte a documentação de TinkerPop atualização.
  - Ao atualizar os drivers da linguagem Gremlin do cliente para Java, observe que as classes do serializador passaram por algumas renomeações. Você precisará atualizar a nomenclatura do pacote e da classe em seus arquivos de configuração e no código, se especificado.
- `StrictTimeoutValidation`(somente quando ativado `StrictTimeoutValidation` por meio do labmode, incluindo`StrictTimeoutValidation=enabled`): quando o `StrictTimeoutValidation` parâmetro tem um valor de`enabled`, um valor de tempo limite por consulta especificado como uma opção de solicitação ou uma dica de consulta não pode exceder o valor definido globalmente no grupo de parâmetros. Nesse caso, Netuno lançará um. `InvalidParameterException` Essa configuração pode ser confirmada em uma resposta no `/status` endpoint quando o valor é e`disabled`, na versão 1.3.2.0 do Neptune, o valor padrão desse parâmetro é. `Disabled`

Melhorias no openCypher

- Consultas de baixa latência e melhoria no desempenho da taxa de transferência: melhorias gerais no desempenho para consultas OpenCypher de baixa latência. A nova versão também melhora a taxa de transferência dessas consultas. As melhorias são mais significativas quando consultas parametrizadas são usadas.
- Support for Query Plan Cache: Quando uma consulta é enviada ao Neptune, a string de consulta é analisada, otimizada e transformada em um plano de consulta, que é executado pelo mecanismo. Os aplicativos geralmente são apoiados por padrões de consulta comuns que são instanciados com valores diferentes. O cache do plano de consulta pode reduzir a latência geral ao armazenar em cache os planos de consulta e, assim, evitar a análise e a otimização desses padrões repetidos.
- Melhoria de desempenho para consultas de agregação DISTINCT.
- Melhoria de desempenho para junções envolvendo variáveis anuláveis.
- Melhoria de desempenho para consultas envolvendo valores diferentes do predicado id (nó/relacionamento).

- Suporte estendido para a funcionalidade de data e hora (ativado apenas no modo `DatetimeMillisecond` de laboratório, incluindo`DatetimeMillisecond=enabled`. Para ter mais informações, consulte Suporte temporal na implementação do Neptune OpenCypher (Neptune Analytics e Neptune Database 1.3.2.0 e versões posteriores).

Defeitos corrigidos nesta versão do motor

Melhorias gerais

- A mensagem de erro do NeptuneML foi atualizada ao validar o acesso aos buckets do Graphlytics.

Correções do Gremlin

- Foram corrigidas as informações de rótulos ausentes na tradução de consultas do DFE, para cenários em que etapas que não contribuem com caminhos contêm rótulos. Por exemplo: .

```
g.withSideEffect('Neptune#useDFE', true).
  V().
  has('name', 'marko').
  has("name", TextP.regex("mark.*")).as("p1").
  not(out().has("name", P.within("peter"))).
  out().as('p2').
  dedup('p1', 'p2')
```

- Corrigido um `NullPointerException` erro na tradução da consulta do DFE, que ocorre quando uma consulta é executada em dois fragmentos do DFE e o primeiro fragmento é otimizado para um nó insatisfatório. Por exemplo: .

```
g.withSideEffect('Neptune#useDFE', true).
  V().
  has('name', 'doesNotExists').
  has("name", TextP.regex("mark.*")).
  inject(1).
  V().
  out().
  has('name', 'vadas')
```

- Corrigido um bug em que Neptune podia lançar `InternalFailureException` um quando uma consulta ValueTraversal continha um modulador by () e sua entrada era Map. Por exemplo: .

```
g.V().
  hasLabel("person").
  project("age", "name").by("age").by("name").
  order().by("age")
```

Correções do openCypher

- Operações UNWIND aprimoradas (por exemplo, expandir uma lista de valores em valores individuais) para ajudar a evitar situações de falta de memória (OOM). Por exemplo: .

```
MATCH (n)-->(m)
WITH collect(m) AS list
UNWIND list AS m
RETURN m, list
```

- Otimização de identificação personalizada corrigida no caso de várias operações MERGE em que a identificação é injetada via UNWIND. Por exemplo: .

```
UNWIND [{nid: 'nid1', mid: 'mid1'}, {nid: 'nid2', mid: 'mid2'}] as ids
MERGE (n:N {`~id`: ids.nid})
MERGE (m:M {`~id`: ids.mid})
```

- Corrigiu a explosão de memória ao planejar consultas complexas com acesso à propriedade e vários saltos com relacionamentos bidirecionais. Por exemplo: .

```
MATCH (person1:person)-[:likes]->(res)-[:partOf]->(group)-[:knows]-(:entity {name:
 'foo'}),
      (person1)-[:knows]->(person2)-[:likes]-(res2), (comment)-[:presentIn]->(:Group
 {name: 'barGroup'}),
      (person1)-[:commented]->(comment2:comment)-[:partOf]->(post:Post), (comment2)-
[:presentIn]->(:Group {name: 'fooGroup'}),
      (comment)-[:contains]->(info:Details)-[:CommentType]->(:CommentType {name:
 'Positive'}),
      (comment2)-[:contains]->(info2:Details)-[:CommentType]->(:CommentType {name:
 'Positive'})
WHERE datetime('2020-01-01T00:00') <= person1.addedAfter <=
 datetime('2023-01-01T23:59') AND comment.approvedBy = comment2.approvedBy
MATCH (comment)-[:contains]->(info3:Details)-[:CommentType]->(:CommentType {name:
 'Neutral'})
RETURN person1, group.name, info1.value,  post.ranking, info3.value
```

- Consultas de agregação fixa com null como grupo por variáveis. Por exemplo: .

```
MATCH (n)
RETURN null AS group, sum(n.num) AS result
```

Correções do SPARQL

- O analisador SPARQL foi corrigido para melhorar o tempo de análise de consultas grandes, como INSERT DATA, contendo muitos triplos e tokens grandes.

Atenuação do problema de cache do plano de consulta

Para a versão 1.3.2.0, detectamos um problema no cache do plano de consulta quando `skip` ou `limit` é usado em uma `WITH` cláusula interna e estamos parametrizados. Por exemplo: .

```
MATCH (n:Person)
WHERE n.age > $age
WITH n skip $skip LIMIT $limit
RETURN n.name, n.age

parameters={"age": 21, "skip": 2, "limit": 3}
```

Nesse caso, os valores dos parâmetros de salto e limite do primeiro plano também serão aplicados às consultas subsequentes, levando a resultados inesperados.

Mitigação

Para evitar esse problema, adicione a dica de consulta `QUERY:PLANCACHE "disabled"` ao enviar uma consulta que inclua uma subcláusula parametrizada de salto e/ou limite. Como alternativa, você pode codificar os valores na consulta.

Opção 1: Usar a dica de consulta para desativar o cache do plano:

```
Using QUERY:PLANCACHE "disabled"
MATCH (n:Person) WHERE n.age > $age
WITH n skip $skip LIMIT $limit
RETURN n.name, n.age

parameters={"age": 21, "skip": 2, "limit": 3}
```

Opção 2: Usar valores codificados para ignorar e limitar:

```
MATCH (n:Person)
WHERE n.age > $age
WITH n skip 2 LIMIT 3
RETURN n.name, n.age

parameters={"age": 21}
```

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.3.2.0, certifique-se de que seu projeto seja compatível com essas versões da linguagem de consulta:

- Versão compatível mais antiga do Gremlin: `3.6.2`
- Versão compatível mais recente do Gremlin: `3.7.1`
- openCypher versão: `Neptune-9.0.20190305-1.0`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.3.2.0 do motor

Você pode atualizar para esta versão a partir da versão 1.2.0.0 ou superior.

Atualizar para esta versão

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.3.2.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.3.2.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Em vez de `--apply-immediately`, é possível especificar `--no-apply-immediately`. Para realizar uma atualização de versão principal, o allow-major-version-upgrade parâmetro é necessário. Além disso, não se esqueça de incluir a versão do mecanismo ou ele poderá ser atualizado para outra versão.

Se o cluster usar um grupo de parâmetros de cluster personalizado, não se esqueça de incluir este parâmetro para especificá-lo:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Da mesma forma, se alguma instância no cluster usar um grupo de parâmetros de banco de dados personalizado, não se esqueça de incluir este parâmetro para especificá-lo:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Se você tiver alguma dúvida ou preocupação, a equipe de AWS Suporte está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Amazon Neptune Engine versão 1.3.1.0 (2024-03-06)

Em 2024-03-06, a versão 1.3.1.0 do motor está sendo geralmente implantada. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

> **Note**
>
> A versão 1.3.0.0 do mecanismo introduziu um novo formato para grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados. Como resultado, se você

estiver atualizando de uma versão de mecanismo anterior à 1.3.0.0 para a versão 1.3.0.0 ou posterior, deverá recriar todos os grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados existentes usando a família de grupos de parâmetros `neptune1.3`. As versões anteriores usavam a família de grupos de parâmetros `neptune1` ou `neptune1.2`, e esses grupos de parâmetros não funcionarão com a versão 1.3.0.0 e posterior. Consulte Grupos de parâmetros do Amazon Neptune Para mais informações.

Melhorias nesta versão do motor

Melhorias gerais

- Neptune melhorou o aviso mostrado em perfil/explique.
- As curvas NIST EC obsoletas foram removidas dos grupos nomeados padrão usados durante a negociação de TLS. As curvas removidas são sect409k1, sect409r1 e sect571k1.

Melhorias no Gremlin

- Computação de estatísticas de DFE aprimorada para evitar NCUs muito altas de instâncias sem servidor.
- Melhoria de desempenho do Gremlin para WITHIN.

Defeitos corrigidos nesta versão do motor

Correções do Gremlin

- Melhorias diversas nos planos de consulta do Gremlin DFE.
- Correção de bug para consultas do Gremlin com uma travessia opcional, por exemplo, para consultas no formato `g.v () .hasLabel ('person') .group () .by (id ()) .by (__.in ('friend') .id () .fold ()) `, onde nenhuma pessoa sem bordas de amizade foi agrupada.
- Corrigido um erro em que consultas do Gremlin contendo etapas de coalescência dentro dos by moduladores faziam com que um erro fosse retornado se executadas usando o mecanismo DFE.
- Corrigido um bug que impedia que consultas somente para leitura executadas em uma sessão do Gremlin funcionassem quando conectadas a uma réplica de leitura.
- Correção de bug em que o ARN do IAM não estava presente no log de auditoria de uma solicitação de conexão inicial bem-sucedida do websocket para o Gremlin.

- Coalesça a etapa, identifique a cobertura da etapa com o DFE.
- Otimização do conjunto de características para planos completos do DFE.

Correções do openCypher

- Correções de bugs na cláusula OpenCypher SET para permitir a configuração em uma expressão não variável (ou seja: match (n:test) set (case when n.prop = 2 then n end) .prop = 3 return n.prop.
- Correção de bug para falhas nas consultas do OpenCypher envolvendo agregação e ordem por.
- UNWIND aprimorado de uma grande lista contendo mapas estáticos.
- Corrija o bug da consulta OpenCypher MERGE usando ID personalizada com valores duplicados.

Correções do SPARQL

- Corrigido um bug do SPARQL sobre o escopo da variável em padrões opcionais.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.3.1.0, certifique-se de que seu projeto seja compatível com essas versões da linguagem de consulta:

- Versão compatível mais antiga do Gremlin: `3.6.2`
- Versão compatível mais recente do Gremlin: `3.6.5`
- openCypher versão: `Neptune-9.0.20190305-1.0`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.3.1.0 do motor

Você pode atualizar para esta versão a partir da versão 1.2.0.0 ou superior.

Atualizar para esta versão

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.3.1.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.3.1.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Em vez de `--apply-immediately`, é possível especificar `--no-apply-immediately`. Para realizar uma atualização de versão principal, o allow-major-version-upgrade parâmetro é necessário. Além disso, não se esqueça de incluir a versão do mecanismo ou ele poderá ser atualizado para outra versão.

Se o cluster usar um grupo de parâmetros de cluster personalizado, não se esqueça de incluir este parâmetro para especificá-lo:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Da mesma forma, se alguma instância no cluster usar um grupo de parâmetros de banco de dados personalizado, não se esqueça de incluir este parâmetro para especificá-lo:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Se você tiver alguma dúvida ou preocupação, a equipe de AWS Suporte está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.3.0.0 (15/11/2023)

Desde 15/11/2023, a versão 1.3.0.0 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

> **Note**
>
> A versão 1.3.0.0 do mecanismo introduziu um novo formato para grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados. Como resultado, se você estiver atualizando de uma versão de mecanismo anterior à 1.3.0.0 para a versão 1.3.0.0 ou posterior, deverá recriar todos os grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados existentes usando a família de grupos de parâmetros `neptune1.3`. As versões anteriores usavam a família de grupos de parâmetros `neptune1` ou `neptune1.2`, e esses grupos de parâmetros não funcionarão com a versão 1.3.0.0 e posterior. Consulte Grupos de parâmetros do Amazon Neptune Para mais informações.

Novos recursos nesta versão do mecanismo

- Lançamento da API de dados do Neptune.

  A API de dados do Amazon Neptune fornece suporte a SDK para mais de 40 operações de dados do Neptune, incluindo carregamento de dados, execução de consultas, consulta de dados e machine learning. Ela é compatível com as três linguagens de consulta (Gremlin, openCypher e SPARQL) e está disponível em todas as linguagens do SDK. Ela assina automaticamente as solicitações da API e simplifica consideravelmente a integração do Neptune às aplicações.
- Foi adicionado suporte para integrar o OpenSearchServerless com o Neptune.

Melhorias nesta versão do mecanismo

Melhorias nas atualizações de mecanismo do Neptune

O Neptune mudou a forma como lança as atualizações do mecanismo para que você possa ter mais controle sobre o processo de atualização. Em vez de lançar patches para alterações ininterruptas, o Neptune agora lança versões secundárias que podem ser controladas usando o campo da instância `AutoMinorVersionUpgrade` e sobre as quais você pode receber notificações ao se inscrever no evento `RDS-EVENT-0156`.

Consulte Maintaining your Amazon Neptune DB Cluster para obter mais informações sobre essas alterações.

Melhoria da criptografia em trânsito

O Neptune não oferece mais suporte aos seguintes pacotes de criptografia:

- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA`

O Neptune oferece suporte apenas aos seguintes conjuntos de criptografia fortes com o TLS 1.2:

- `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256`

Melhorias no Gremlin

- Adição de suporte ao mecanismo do DFE para as seguintes etapas do Gremlin:
  - `FoldStep`
  - `GroupStep`
  - `GroupCountStep`
  - `TraversalMapStep`
  - `UnfoldStep`
  - `LabelStep`
  - `PropertyKeyStep`
  - `PropertyValueStep`
  - `AndStep`
  - `OrStep`
  - `ConstantStep`
  - `CountGlobalStep`
- Planos de consulta do DFE do Gremlin otimizados para evitar varreduras completas de vértices ao usar a modulação `by()`.

- Desempenho significativamente aprimorado de consultas de baixa cardinalidade e baixa latência.
- Foi adicionado suporte ao DFE para predicados TinkerPop `0r` de filtro.
- Suporte aprimorado de percurso do DFE para filtros na mesma chave, para consultas como as seguintes:

```
g.withSideEffect("Neptune#useDFE", true)
 .V()
 .has('name', 'marko')
 .and(
   or(
     has('name', eq("marko")),
     has('name', eq("vardas"))
   )
 )
```

- Tratamento aprimorado de erros para a etapa `fail()`.

Melhorias no openCypher

- Desempenho significativamente aprimorado de consultas de baixa cardinalidade e baixa latência.
- Melhor desempenho do planejamento de consultas quando a consulta contém muitos tipos de nós.
- Latência reduzida de todas as consultas VLP.
- Desempenho aprimorado com a remoção de junções de pipeline redundantes para consultas de padrões de nó único.
- Desempenho aprimorado para consultas que contêm padrões de vários saltos com ciclos, como esta:

```
MATCH (n)-->()-->()-->(m)
RETURN n m
```

Melhorias no SPARQL

- Introdução de um novo operador SPARQL: `PipelineHashIndexJoin`.
- Melhor desempenho da validação de URI para consultas SPARQL.
- Melhor desempenho das consultas de pesquisa de texto completo do SPARQL por meio da resolução em lote de termos do dicionário.

Defeitos corrigidos nesta versão do mecanismo

Correções do Gremlin

- Correção de um erro do Gremlin em que ocorria um vazamento de transação ao conferir o endpoint de status da consulta do Gremlin para consultas com predicados em percursos secundários para etapas não processadas de modo nativo no mecanismo do DFE.
- Correção de um erro do Gremlin que fazia com que `valueMap()` não fosse otimizado no mecanismo do DFE em `by()`.
- Correção de um erro do Gremlin que fazia com que um rótulo de etapa anexado a `UnionStep` não fosse propagado para o último elemento do caminho dos respectivos percursos secundários.
- Corrigido um bug do Gremlin em que uma consulta falhava porque continha muitas TinkerPop etapas e, em seguida, não era limpa.
- Correção de um erro do Gremlin em que `NullPointerException` era lançado nas etapas `mergeV` e `mergeE`.
- Correção de um erro do Gremlin em que `order()` não classificava corretamente as saídas de string quando algumas delas continham um caractere de espaço.
- Correção de um problema de correção do Gremlin que ocorria quando a etapa `valueMap` era processada no mecanismo do DFE.
- Correção de um problema de correção do Gremlin que ocorria quando `GroupStep` ou `GroupCountStep` estava aninhado em um percurso de chaves.

Correções do openCypher

- Correção de um problema do openCypher envolvendo o tratamento de erros em torno de caracteres NULL.
- Correção de um erro do openCypher no tratamento de transações do Bolt.

Correções do SPARQL

- Correção de um erro do SPARQL em que os valores dentro de funções recursivas não eram resolvidos adequadamente.
- Correção de um erro do SPARQL que causava degradação do desempenho quando um grande número de valores eram injetados usando uma cláusula `VALUES`.

- Correção de um erro do SPARQL em que uma chamada para o operador `REGEX` em um literal marcado com uma linguagem nunca tinha êxito.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.3.0.0, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Versão compatível mais antiga do Gremlin: `3.6.2`
- Versão compatível mais recente do Gremlin: `3.6.4`
- openCypher versão: `Neptune-9.0.20190305-1.0`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.3.0.0 do mecanismo

Você pode atualizar para esta versão a partir da versão 1.2.0.0 ou superior.

Atualizar para esta versão

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.3.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.3.0.0 ^
    --allow-major-version-upgrade ^
```

```
    --apply-immediately
```

Em vez de `--apply-immediately`, é possível especificar `--no-apply-immediately`. Para realizar uma atualização de versão principal, o allow-major-version-upgrade parâmetro é necessário. Além disso, não se esqueça de incluir a versão do mecanismo ou ele poderá ser atualizado para outra versão.

Se o cluster usar um grupo de parâmetros de cluster personalizado, não se esqueça de incluir este parâmetro para especificá-lo:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Da mesma forma, se alguma instância no cluster usar um grupo de parâmetros de banco de dados personalizado, não se esqueça de incluir este parâmetro para especificá-lo:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot

manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Se você tiver alguma dúvida ou preocupação, a equipe de AWS Suporte está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Amazon Neptune Engine versão 1.2.1.1 (2024-03-11)

Em 11/03/2024/03, a versão 1.2.1.1 do mecanismo geralmente está sendo implantada. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

> **Note**
>
> Se estiver fazendo a atualização de uma versão do mecanismo anterior à 1.2.0.0:
>
> - A versão 1.2.0.0 do mecanismo introduziu um novo formato para grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados. Como resultado, se você estiver atualizando de uma versão de mecanismo anterior à 1.2.0.0 para a versão 1.2.0.0 ou posterior, deverá recriar todos os grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados existentes usando a família de grupos

de parâmetros `neptune1.2`. As versões anteriores usavam a família de grupos de parâmetros `neptune1`, e esses grupos de parâmetros não funcionarão com a versão 1.2.0.0 e posterior. Consulte Grupos de parâmetros do Amazon Neptune Para mais informações.

- A versão 1.2.0.0 do mecanismo também introduziu um novo formato para undo logs. Como resultado, todos os registros de desfazer criados por uma versão anterior do mecanismo devem ser eliminados e a `UndoLogsListSize` CloudWatch métrica deve cair para zero antes que qualquer atualização de uma versão anterior à 1.2.0.0 possa ser iniciada. Se houver muitos registros de undo logs (duzentos mil ou mais) ao tentar iniciar uma atualização, a tentativa de atualização poderá expirar enquanto aguarda a conclusão da limpeza dos undo logs.

  É possível acelerar a taxa de limpeza atualizando a instância de gravador do cluster, que é onde a limpeza ocorre. Fazer isso antes de tentar realizar a atualização pode reduzir o número de undo logs antes de começar. Aumentar o tamanho do gravador para um tipo de instância 24XL pode aumentar a taxa de limpeza para mais de um milhão de registros por hora.

  Se a `UndoLogsListSize` CloudWatch métrica for extremamente grande, abrir um caso de suporte pode ajudá-lo a explorar estratégias adicionais para reduzi-la.
- Por fim, houve uma alteração significativa na versão 1.2.0.0. Ela afeta o código anterior que usava o protocolo Bolt com autenticação do IAM. A partir da versão 1.2.0.0, o Bolt precisa de um caminho de recursos para a assinatura do IAM. Em Java, a definição do caminho de recursos pode ser assim: `request.setResourcePath("/openCypher"));`. Em outras linguagens, o `/openCypher` pode ser anexado ao URI do endpoint. Consulte Usar o protocolo Bolt para ver exemplos.

Melhorias nesta versão do mecanismo

Melhorias gerais

Neptune melhorou o aviso mostrado em perfil/explique.

Melhorias no Gremlin

- Computação de estatísticas de DFE aprimorada para evitar NCUs muito altas de instâncias sem servidor.

- Melhoria de desempenho do Gremlin para WITHIN.

Defeitos corrigidos nesta versão do mecanismo

Correções do Gremlin

- Correções de bugs com a ordenação do plano de consulta do mecanismo Gremlin DFE.
- Correção de bug com out-of-memory erro Gremlin quando originalmente relatado como. InternalFailureException
- Correção de bug em que o ARN do IAM não estava presente no log de auditoria de uma solicitação de conexão inicial bem-sucedida do websocket.
- Correção de bug para consultas do Gremlin com TinkerPop sessão ativada quando as consultas em uma sessão falham, mesmo quando todas elas são somente de leitura e conectadas a uma instância de leitura.

Correções do openCypher

- Correções de bugs na cláusula OpenCypher SET para permitir a configuração em uma expressão não variável (ou seja: match (n:test) set (case when n.prop = 2 then n end) .prop = 3 return n.prop.
- Correção de bug para falhas nas consultas do OpenCypher envolvendo agregação e ordem por.
- UNWIND aprimorado de uma grande lista contendo mapas estáticos.
- Corrija o bug da consulta OpenCypher MERGE usando ID personalizada com valores duplicados.

Correções do SPARQL

- Correções de erros no planejador de consultas SPARQL DFE.
- Correção de erros para SPARQL quando usado com as palavras-chave BIND e OPTIONAL.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.2.1.1, certifique-se de que seu projeto seja compatível com essas versões da linguagem de consulta:

- Versão compatível mais antiga do Gremlin: `3.6.2`
- Versão compatível mais recente do Gremlin: `3.6.2`

- openCypher versão: `Neptune-9.0.20190305-1.0`
- SPARQL versão: `1.1`

Caminhos de atualização para o Engine Release 1.2.1.1

Você pode atualizar para esta versão a partir da versão 1.2.0.0 ou superior.

Atualizar para esta versão

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.1 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.1 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Em vez de `--apply-immediately`, é possível especificar `--no-apply-immediately`. Para realizar uma atualização de versão principal, o allow-major-version-upgrade parâmetro é necessário. Além disso, não se esqueça de incluir a versão do mecanismo ou ele poderá ser atualizado para outra versão.

Se o cluster usar um grupo de parâmetros de cluster personalizado, não se esqueça de incluir este parâmetro para especificá-lo:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Da mesma forma, se alguma instância no cluster usar um grupo de parâmetros de banco de dados personalizado, não se esqueça de incluir este parâmetro para especificá-lo:

```
--db-instance-parameter-group-name (name of the custom instance parameter group)
```

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> ```

```
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Se você tiver alguma dúvida ou preocupação, a equipe de AWS Suporte está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.2.1.0 (08/03/2023)

Desde 08/03/2023, a versão 1.2.1.0 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Note

Se estiver fazendo a atualização de uma versão do mecanismo anterior à 1.2.0.0:

- A versão 1.2.0.0 do mecanismo introduziu um novo formato para grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados. Como resultado, se você estiver atualizando de uma versão de mecanismo anterior à 1.2.0.0 para a versão 1.2.0.0 ou posterior, deverá recriar todos os grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados existentes usando a família de grupos de parâmetros `neptune1.2`. As versões anteriores usavam a família de grupos de parâmetros `neptune1`, e esses grupos de parâmetros não funcionarão com a versão 1.2.0.0 e posterior. Consulte Grupos de parâmetros do Amazon Neptune para obter mais informações.
- A versão 1.2.0.0 do mecanismo também introduziu um novo formato para undo logs. Como resultado, todos os undo logs criados por uma versão anterior do mecanismo devem ser eliminados e a métrica `UndoLogsListSize` do CloudWatch deve cair para zero para que seja possível iniciar qualquer atualização de uma versão anterior à 1.2.0.0. Se houver muitos registros de undo logs (duzentos mil ou mais) ao tentar iniciar uma atualização, a tentativa de atualização poderá expirar enquanto aguarda a conclusão da limpeza dos undo logs.

É possível acelerar a taxa de limpeza atualizando a instância de gravador do cluster, que é onde a limpeza ocorre. Fazer isso antes de tentar realizar a atualização pode reduzir o número de undo logs antes de começar. Aumentar o tamanho do gravador para um tipo de instância 24XL pode aumentar a taxa de limpeza para mais de um milhão de registros por hora.

Se a métrica `UndoLogsListSize` do CloudWatch for extremamente grande, abrir um caso de suporte pode ajudar a examinar estratégias adicionais para reduzi-la.

- Por fim, houve uma alteração significativa na versão 1.2.0.0. Ela afeta o código anterior que usava o protocolo Bolt com autenticação do IAM. A partir da versão 1.2.0.0, o Bolt precisa de um caminho de recursos para a assinatura do IAM. Em Java, a definição do caminho de recursos pode ser assim: `request.setResourcePath("/openCypher"));`. Em outras linguagens, o `/openCypher` pode ser anexado ao URI do endpoint. Consulte Usar o protocolo Bolt para ver exemplos.

Versões de patch subsequentes para esta versão

- Versão: 1.2.1.0.R2 (02/05/2023)
- Versão: 1.2.1.0.R3 (13/06/2023)
- Versão: 1.2.1.0.R4 (10/08/2023)
- Versão: 1.2.1.0.R5 (02/09/2023)
- Versão: 1.2.1.0.R6 (12/09/2023)
- Versão: 1.2.1.0.R7 (06/10/2023)

Novos recursos nesta versão do mecanismo

- Adição de suporte para o TinkerPop 3.6.2, que adiciona vários novos atributos do Gremlin, como as novas etapas `mergeV()`, `mergeE()`, `element()` e `fail()`. As etapas `mergeV()` e `mergeE()` são particularmente importantes, pois oferecem uma opção declarativa há muito esperada para realizar operações do tipo upsert, o que deve simplificar bastante os padrões de código existentes e facilitar a leitura do Gremlin. A versão 3.6.x também adicionou predicados regex, uma nova sobrecarga à etapa `property()` que precisa de uma `Map` e uma grande revisão do comportamento da modulação `by()` que é muito mais consistente em todas as etapas que a utilizam.

Consulte a TinkerPop change log e de atualização para obter informações sobre as alterações na versão 3.6 e fatores a serem considerados ao realizar a atualização.

Se você estiver usando `fold().coalesce(unfold(), <mutate>)` para inserções condicionais, recomendamos migrar para a nova sintaxe `mergeV/E()`, descrita aqui e aqui. O Neptune usa um padrão de bloqueio mais estreito para `Merge` do que para `Coalesce`, o que pode reduzir as exceções de modificação simultânea (CMEs).

Para obter mais informações sobre os novos atributos disponíveis nesta versão do TinkerPop, consulte o blog de Stephen Mallette, Exploring new features of Apache TinkerPop 3.6.x in Amazon Neptune.

- Adição de suporte para tipos de instância R6i, equipados com processadores escaláveis Intel Xeon de terceira geração. Eles são ideais para workloads com uso intenso de memória e oferecem relação de computação/preço até 15% melhor e largura de banda de memória até 20% maior por vCPU do que tipos de instância R5 comparáveis.
- Adição de endpoints de API de resumo de grafos para grafos de propriedades e do RDF, que permitem que você obtenha um relatório resumido rápido sobre o grafo.

  Para grafos de propriedades (PG), a API de resumo de grafos gera uma lista somente leitura de rótulos de nós e bordas e chaves de propriedade, junto com contagens de nós, bordas e propriedades. Para grafos do RDF, ela fornece uma lista de classes e chaves de predicados, junto com contagens de quadrantes, assuntos e predicados.

  As seguintes alterações foram realizadas com nova API de resumo de grafos:

  - Adição de uma nova ação de plano de dados GetGraphSummary.
  - Adição de um novo endpoint `rdf/statistics` para substituir o endpoint `sparql/statistics` que agora está obsoleto.
  - Alteração do nome do campo `summary` na resposta do status das estatísticas para `signatureInfo`, para não confundi-lo com as informações de resumo dos grafos. As versões anteriores do mecanismo continuam usando `summary` na resposta JSON.
  - Alteração da precisão do campo `date` na resposta do status das estatísticas de minuto para milissegundo. O formato anterior era `2020-05-07T23:13Z` (precisão de minutos), enquanto o novo formato é `2023-01-24T00:47:43.319Z` (precisão de milissegundos). Os dois são compatíveis com a ISO 8601, mas essa alteração pode romper o código existente, dependendo de como a data está sendo analisada.

  - Adição de uma nova magia de linha `%statistics` à bancada de trabalho que permite recuperar estatísticas do mecanismo do DFE.
  - Adição de uma nova magia de linha `%summary` à bancada de trabalho que permite recuperar informações de resumo dos grafos.
- Adição do registro em log de consultas lentas para registrar consultas que demoram mais do que um limite especificado para serem executadas. Você ativa e controla o registro em log de consultas lentas usando os dois novos parâmetros dinâmicos, a saber, neptune_enable_slow_query_log e neptune_slow_query_log_threshold.
- Adição de suporte para dois parâmetros dinâmicos, a saber, os novos parâmetros de cluster, neptune_enable_slow_query_log e neptune_slow_query_log_threshold. Ao fazer uma alteração em um parâmetro dinâmico, ela tem efeito imediatamente, sem a necessidade de reinicializar a instância.
- Adição de uma função removeKeyFromMap() do openCypher específica do Neptune que remove uma chave especificada de um mapa e exibe o novo mapa resultante.

Melhorias nesta versão do mecanismo

- Extensão do suporte ao DFE do Gremlin para etapas `limit` com escopo local.
- Adição de suporte de modulação `by()` para `DedupGlobalStep` do Gremlin no mecanismo do DFE.
- Adição de suporte do DFE para `SelectStep` e `SelectOneStep` do Gremlin.
- Melhorias no desempenho e correções para vários operadores do Gremlin, incluindo `repeat`, `coalesce`, `store` e `aggregate`.
- Melhoria no desempenho das consultas do openCypher que envolvem `MERGE` e `OPTIONAL MATCH`.
- Melhoria no desempenho das consultas do openCypher que envolvem `UNWIND` de uma lista de mapas de valores literais.
- Melhoria no desempenho das consultas do openCypher que têm um filtro `IN` para `id`. Por exemplo:

```
MATCH (n) WHERE id(n) IN ['1', '2', '3'] RETURN n
```

- Adição da capacidade de especificar o IRI básico para consultas do SPARQL usando a declaração BASE (consulte IRI de base padrão para consultas e atualizações).

- Redução do tempo de espera do processamento de carga para carregamentos em massa somente de bordas do Gremlin e do openCypher.
- Os carregamentos em massa agora são retomados de forma assíncrona quando o Neptune é reiniciado para evitar um longo tempo de espera causado por problemas de conectividade do Amazon S3 antes de falhar nas tentativas de retomada.
- Melhoria no tratamento das consultas SPARQL DESCRIBE que têm a dica de consulta describeMode definida como `"CBD"` (descrição concisa limitada) e que envolvem um grande número de nós em branco.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro do openCypher em que as consultas geravam uma string, `"null"`, em vez de um valor nulo em Bolt e SPARQL-JSON.
- Correção de um erro do openCypher na compreensão de lista que produzia um valor nulo em vez dos valores fornecidos para os elementos da lista.
- Correção de um erro do openCypher em que os valores de bytes não eram serializados corretamente.
- Correção de um erro do Gremlin em `UnionStep` que ocorria quando uma entrada era uma borda atravessando um vértice em um percurso secundário.
- Correção de um erro do Gremlin que fazia com que um rótulo de etapa associado a `UnionStep` não se propagasse corretamente até a última etapa de cada percurso secundário.
- Correção de um erro do Gremlin na etapa `dedup` com rótulos após uma etapa `repeat`, em que os rótulos anexados à etapa `dedup` não estavam disponíveis para uso posterior na consulta.
- Correção de um erro do Gremlin em que a conversão da etapa `repeat` dentro de uma etapa `union` falhava com um erro interno.
- Correção dos problemas de exatidão do Gremlin para consultas do DFE com `limit` como percurso filho de etapas que não são de junção retornando ao Tinkerpop. Consultas em um formato como este são afetadas:

```
g.withSideEffect('Neptune#useDFE', true).V().as("a").select("a").by(out().limit(1))
```

- Correção de um erro do SPARQL em que os padrões `SPARQL GRAPH` não consideravam o conjunto de dados fornecido por uma cláusula `FROM NAMED`.

- Correção de um erro do SPARQL em que `DESCRIBE` do SPARQL com algumas cláusulas `FROM` e/ou `FROM NAMED` nem sempre usava corretamente os dados de grafos padrão e às vezes gerava uma exceção. Consulte Comportamento do SPARQL DESCRIBE em relação ao grafo padrão.
- Correção de um erro do SPARQL para que a mensagem de exceção correta fosse exibida quando caracteres nulos fossem rejeitados.
- Correção de um erro de explain do SPARQL que afetava os planos que continham um operador PipelinedHashIndexJoin.
- Correção de um erro que causava um erro interno quando uma consulta que gerava um valor constante era enviada.
- Correção de um problema com a lógica do detector de deadlock que ocasionalmente fazia com que o mecanismo parasse de responder.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.2.1.0, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Versão compatível mais antiga do Gremlin: `3.6.2`
- Versão compatível mais recente do Gremlin: `3.6.2`
- openCypher versão: `Neptune-9.0.20190305-1.1`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.2.1.0 do mecanismo

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune maior ou igual a 1.1.0.0 para esta versão.

> **Note**
>
> A partir da versão 1.2.0.0 do mecanismo, todos os grupos de parâmetros personalizados e os grupos de parâmetros de cluster personalizados utilizados com versões do mecanismo anteriores à `1.2.0.0` agora devem ser recriados usando a família de grupos de parâmetros `neptune1.2`. As versões anteriores usavam a família de grupos de parâmetros `neptune1`, e esses grupos de parâmetros não funcionarão com versões a partir de `1.2.0.0` em diante. Consulte Grupos de parâmetros do Amazon Neptune para obter mais informações.

A atualização para esta versão principal não será automática.

Atualizar para esta versão

O Amazon Neptune 1.2.1.0 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.2.1.0.R7 (06/10/2023)

Desde 06/10/2023, a versão 1.2.1.0.R7 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Note

Se estiver fazendo a atualização de uma versão do mecanismo anterior à 1.2.0.0:

- A versão 1.2.0.0 do mecanismo introduziu um novo formato para grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados. Como resultado, se você estiver atualizando de uma versão de mecanismo anterior à 1.2.0.0 para a versão 1.2.0.0 ou posterior, deverá recriar todos os grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados existentes usando a família de grupos de parâmetros `neptune1.2`. As versões anteriores usavam a família de grupos de parâmetros `neptune1`, e esses grupos de parâmetros não funcionarão com a versão 1.2.0.0 e posterior. Consulte Grupos de parâmetros do Amazon Neptune para obter mais informações.
- A versão 1.2.0.0 do mecanismo também introduziu um novo formato para undo logs. Como resultado, todos os undo logs criados por uma versão anterior do mecanismo devem ser eliminados e a métrica `UndoLogsListSize` do CloudWatch deve cair para zero para que seja possível iniciar qualquer atualização de uma versão anterior à 1.2.0.0. Se houver muitos registros de undo logs (duzentos mil ou mais) ao tentar iniciar uma atualização, a tentativa de atualização poderá expirar enquanto aguarda a conclusão da limpeza dos undo logs.

  É possível acelerar a taxa de limpeza atualizando a instância de gravador do cluster, que é onde a limpeza ocorre. Fazer isso antes de tentar realizar a atualização pode reduzir o número de undo logs antes de começar. Aumentar o tamanho do gravador para um tipo de instância 24XL pode aumentar a taxa de limpeza para mais de um milhão de registros por hora.

  Se a métrica `UndoLogsListSize` do CloudWatch for extremamente grande, abrir um caso de suporte pode ajudar a examinar estratégias adicionais para reduzi-la.
- Por fim, houve uma alteração significativa na versão 1.2.0.0. Ela afeta o código anterior que usava o protocolo Bolt com autenticação do IAM. A partir da versão 1.2.0.0, o Bolt precisa de um caminho de recursos para a assinatura do IAM. Em Java, a definição do caminho de recursos pode ser assim: `request.setResourcePath("/`

`openCypher"));`. Em outras linguagens, o `/openCypher` pode ser anexado ao URI do endpoint. Consulte Usar o protocolo Bolt para ver exemplos.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro em que, em alguns casos, uma transação com falha não era fechada corretamente.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.2.1.0.R7, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Versão compatível mais antiga do Gremlin: `3.6.2`
- Versão compatível mais recente do Gremlin: `3.6.2`
- openCypher versão: `Neptune-9.0.20190305-1.0`
- SPARQL versão: `1.1`

Atualizar para esta versão

O Amazon Neptune 1.2.1.0.R7 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
```

```
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.2.1.0.R6 (12/09/2023)

Desde 12/09/2023, a versão 1.2.1.0.R6 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Note

Se estiver fazendo a atualização de uma versão do mecanismo anterior à 1.2.0.0:

- A versão 1.2.0.0 do mecanismo introduziu um novo formato para grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados. Como resultado, se você estiver atualizando de uma versão de mecanismo anterior à 1.2.0.0 para a versão 1.2.0.0 ou posterior, deverá recriar todos os grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados existentes usando a família de grupos de parâmetros `neptune1.2`. As versões anteriores usavam a família de grupos de parâmetros `neptune1`, e esses grupos de parâmetros não funcionarão com a versão 1.2.0.0 e posterior. Consulte Grupos de parâmetros do Amazon Neptune para obter mais informações.
- A versão 1.2.0.0 do mecanismo também introduziu um novo formato para undo logs. Como resultado, todos os undo logs criados por uma versão anterior do mecanismo devem ser eliminados e a métrica `UndoLogsListSize` do CloudWatch deve cair para zero para que seja possível iniciar qualquer atualização de uma versão anterior à 1.2.0.0. Se houver

muitos registros de undo logs (duzentos mil ou mais) ao tentar iniciar uma atualização, a tentativa de atualização poderá expirar enquanto aguarda a conclusão da limpeza dos undo logs.

É possível acelerar a taxa de limpeza atualizando a instância de gravador do cluster, que é onde a limpeza ocorre. Fazer isso antes de tentar realizar a atualização pode reduzir o número de undo logs antes de começar. Aumentar o tamanho do gravador para um tipo de instância 24XL pode aumentar a taxa de limpeza para mais de um milhão de registros por hora.

Se a métrica `UndoLogsListSize` do CloudWatch for extremamente grande, abrir um caso de suporte pode ajudar a examinar estratégias adicionais para reduzi-la.

- Por fim, houve uma alteração significativa na versão 1.2.0.0. Ela afeta o código anterior que usava o protocolo Bolt com autenticação do IAM. A partir da versão 1.2.0.0, o Bolt precisa de um caminho de recursos para a assinatura do IAM. Em Java, a definição do caminho de recursos pode ser assim: `request.setResourcePath("/openCypher"));`. Em outras linguagens, o `/openCypher` pode ser anexado ao URI do endpoint. Consulte Usar o protocolo Bolt para ver exemplos.

Novos recursos nesta versão do mecanismo

- Lançamento da API de dados do Neptune.

  A API de dados do Amazon Neptune é compatível com o SDK para carregar dados, executar consultas, obter informações sobre os dados e executar operações de machine learning. Ela é compatível com as linguagens de consulta Gremlin e openCypher no Neptune e está disponível em todas as linguagens do SDK. Ela assina automaticamente as solicitações da API e simplifica consideravelmente a integração do Neptune às aplicações.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro que poderia causar picos de CPU sob altas cargas quando os logs de consultas lentas eram habilitados.
- Correção de um erro do Gremlin em que adicionar uma borda e suas propriedades seguidas por `inV()` ou `outV()` gerava uma `InternalFailureException`.

- Correção de alguns problemas com o encadeamento de perfis do IAM que causavam degradação no desempenho do carregador em massa em alguns casos.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.2.1.0.R6, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Versão compatível mais antiga do Gremlin: `3.6.2`
- Versão compatível mais recente do Gremlin: `3.6.2`
- openCypher versão: `Neptune-9.0.20190305-1.0`
- SPARQL versão: `1.1`

Atualizar para esta versão

O Amazon Neptune 1.2.1.0.R6 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias,

ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the
> instance before
> ```

```
proceeding with the upgrade.
```

Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.2.1.0.R5 (02/09/2023)

Desde 02/09/2023, a versão 1.2.1.0.R5 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Note

Se estiver fazendo a atualização de uma versão do mecanismo anterior à 1.2.0.0:

- A versão 1.2.0.0 do mecanismo introduziu um novo formato para grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados. Como resultado, se você estiver atualizando de uma versão de mecanismo anterior à 1.2.0.0 para a versão 1.2.0.0 ou posterior, deverá recriar todos os grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados existentes usando a família de grupos de parâmetros `neptune1.2`. As versões anteriores usavam a família de grupos de parâmetros `neptune1`, e esses grupos de parâmetros não funcionarão com a versão 1.2.0.0 e posterior. Consulte Grupos de parâmetros do Amazon Neptune para obter mais informações.
- A versão 1.2.0.0 do mecanismo também introduziu um novo formato para undo logs. Como resultado, todos os undo logs criados por uma versão anterior do mecanismo devem ser eliminados e a métrica `UndoLogsListSize` do CloudWatch deve cair para zero para que seja possível iniciar qualquer atualização de uma versão anterior à 1.2.0.0. Se houver muitos registros de undo logs (duzentos mil ou mais) ao tentar iniciar uma atualização, a tentativa de atualização poderá expirar enquanto aguarda a conclusão da limpeza dos undo logs.

É possível acelerar a taxa de limpeza atualizando a instância de gravador do cluster, que é onde a limpeza ocorre. Fazer isso antes de tentar realizar a atualização pode reduzir o número de undo logs antes de começar. Aumentar o tamanho do gravador para um tipo de instância 24XL pode aumentar a taxa de limpeza para mais de um milhão de registros por hora.

Se a métrica `UndoLogsListSize` do CloudWatch for extremamente grande, abrir um caso de suporte pode ajudar a examinar estratégias adicionais para reduzi-la.

- Por fim, houve uma alteração significativa na versão 1.2.0.0. Ela afeta o código anterior que usava o protocolo Bolt com autenticação do IAM. A partir da versão 1.2.0.0, o Bolt precisa de um caminho de recursos para a assinatura do IAM. Em Java, a definição do caminho de recursos pode ser assim: `request.setResourcePath("/openCypher"));`. Em outras linguagens, o `/openCypher` pode ser anexado ao URI do endpoint. Consulte Usar o protocolo Bolt para ver exemplos.

Novos recursos nesta versão do mecanismo

- Lançamento da API de dados do Neptune.

  A API de dados do Amazon Neptune é compatível com o SDK para carregar dados, executar consultas, obter informações sobre os dados e executar operações de machine learning. Ela é compatível com as linguagens de consulta Gremlin e openCypher no Neptune e está disponível em todas as linguagens do SDK. Ela assina automaticamente as solicitações da API e simplifica consideravelmente a integração do Neptune às aplicações.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro do Gremlin em que adicionar uma borda e suas propriedades seguidas por `inV()` ou `outV()` gerava uma `InternalFailureException`.
- Correção de alguns problemas com o encadeamento de perfis do IAM que causavam degradação no desempenho do carregador em massa em alguns casos.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.2.1.0.R5, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Versão compatível mais antiga do Gremlin: `3.6.2`
- Versão compatível mais recente do Gremlin: `3.6.2`
- openCypher versão: `Neptune-9.0.20190305-1.0`
- SPARQL versão: `1.1`

Atualizar para esta versão

O Amazon Neptune 1.2.1.0.R5 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```

Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.2.1.0.R4 (10/08/2023)

Desde 10/08/2023, a versão 1.2.1.0.R4 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Important

As alterações introduzidas nesta versão do mecanismo, em alguns casos, podem ocasionar degradação do desempenho do carregamento em massa. Como resultado, as atualizações para esta versão foram temporariamente suspensas até que o problema seja resolvido.

Note

Se estiver fazendo a atualização de uma versão do mecanismo anterior à 1.2.0.0:

- A versão 1.2.0.0 do mecanismo introduziu um novo formato para grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados. Como resultado, se você estiver atualizando de uma versão de mecanismo anterior à 1.2.0.0 para a versão 1.2.0.0 ou posterior, deverá recriar todos os grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados existentes usando a família de grupos de parâmetros `neptune1.2`. As versões anteriores usavam a família de grupos de parâmetros `neptune1`, e esses grupos de parâmetros não funcionarão com a versão 1.2.0.0 e posterior. Consulte Grupos de parâmetros do Amazon Neptune para obter mais informações.
- A versão 1.2.0.0 do mecanismo também introduziu um novo formato para undo logs. Como resultado, todos os undo logs criados por uma versão anterior do mecanismo devem ser eliminados e a métrica `UndoLogsListSize` do CloudWatch deve cair para zero para

que seja possível iniciar qualquer atualização de uma versão anterior à 1.2.0.0. Se houver muitos registros de undo logs (duzentos mil ou mais) ao tentar iniciar uma atualização, a tentativa de atualização poderá expirar enquanto aguarda a conclusão da limpeza dos undo logs.

É possível acelerar a taxa de limpeza atualizando a instância de gravador do cluster, que é onde a limpeza ocorre. Fazer isso antes de tentar realizar a atualização pode reduzir o número de undo logs antes de começar. Aumentar o tamanho do gravador para um tipo de instância 24XL pode aumentar a taxa de limpeza para mais de um milhão de registros por hora.

Se a métrica `UndoLogsListSize` do CloudWatch for extremamente grande, abrir um caso de suporte pode ajudar a examinar estratégias adicionais para reduzi-la.

- Por fim, houve uma alteração significativa na versão 1.2.0.0. Ela afeta o código anterior que usava o protocolo Bolt com autenticação do IAM. A partir da versão 1.2.0.0, o Bolt precisa de um caminho de recursos para a assinatura do IAM. Em Java, a definição do caminho de recursos pode ser assim: `request.setResourcePath("/openCypher"));`. Em outras linguagens, o `/openCypher` pode ser anexado ao URI do endpoint. Consulte Usar o protocolo Bolt para ver exemplos.

Melhorias nesta versão do mecanismo

- Adição de suporte ao GraphSON-1.0 para Gremlin. Para usar o GraphSON-1.0, transmita `Accept header` com um valor de:

```
application/vnd.gremlin-v1.0+json;types=false
```

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro do Gremlin em que ocorria um vazamento de transação ao conferir o endpoint de status da consulta do Gremlin para consultas com predicados em percursos secundários para etapas não processadas de modo nativo.
- Correção de um erro do openCypher no tratamento de transações do Bolt.
- Correção de um problema de simultaneidade na camada de armazenamento que poderia causar uma falha.

- Correção de um erro nos logs de consultas lentas para garantir que eles não estejam ativos quando desabilitados.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.2.1.0.R4, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Versão compatível mais antiga do Gremlin: `3.6.2`
- Versão compatível mais recente do Gremlin: `3.6.5`
- openCypher versão: `Neptune-9.0.20190305-1.0`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.2.1.0.R4 do mecanismo

Atualizar para esta versão

O Amazon Neptune 1.2.1.0.R4 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
> ```

```
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.2.1.0.R3 (13/06/2023)

Desde 13/06/2023, a versão 1.2.1.0.R3 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Important

As alterações introduzidas nesta versão do mecanismo, em alguns casos, podem ocasionar degradação do desempenho do carregamento em massa. Como resultado, as atualizações para esta versão foram temporariamente suspensas até que o problema seja resolvido.

Note

Se estiver fazendo a atualização de uma versão do mecanismo anterior à 1.2.0.0:

- A versão 1.2.0.0 do mecanismo introduziu um novo formato para grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados. Como resultado, se você estiver atualizando de uma versão de mecanismo anterior à 1.2.0.0 para a versão 1.2.0.0 ou posterior, deverá recriar todos os grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados existentes usando a família de grupos de parâmetros `neptune1.2`. As versões anteriores usavam a família de grupos de parâmetros `neptune1`, e esses grupos de parâmetros não funcionarão com a versão 1.2.0.0 e posterior. Consulte Grupos de parâmetros do Amazon Neptune para obter mais informações.

- A versão 1.2.0.0 do mecanismo também introduziu um novo formato para undo logs. Como resultado, todos os undo logs criados por uma versão anterior do mecanismo devem ser eliminados e a métrica `UndoLogsListSize` do CloudWatch deve cair para zero para que seja possível iniciar qualquer atualização de uma versão anterior à 1.2.0.0. Se houver muitos registros de undo logs (duzentos mil ou mais) ao tentar iniciar uma atualização, a tentativa de atualização poderá expirar enquanto aguarda a conclusão da limpeza dos undo logs.

  É possível acelerar a taxa de limpeza atualizando a instância de gravador do cluster, que é onde a limpeza ocorre. Fazer isso antes de tentar realizar a atualização pode reduzir o número de undo logs antes de começar. Aumentar o tamanho do gravador para um tipo de instância 24XL pode aumentar a taxa de limpeza para mais de um milhão de registros por hora.

  Se a métrica `UndoLogsListSize` do CloudWatch for extremamente grande, abrir um caso de suporte pode ajudar a examinar estratégias adicionais para reduzi-la.
- Por fim, houve uma alteração significativa na versão 1.2.0.0. Ela afeta o código anterior que usava o protocolo Bolt com autenticação do IAM. A partir da versão 1.2.0.0, o Bolt precisa de um caminho de recursos para a assinatura do IAM. Em Java, a definição do caminho de recursos pode ser assim: `request.setResourcePath("/openCypher"));`. Em outras linguagens, o `/openCypher` pode ser anexado ao URI do endpoint. Consulte Usar o protocolo Bolt para ver exemplos.

Novos recursos nesta versão do mecanismo

- Adição de suporte para carregamento em massa entre contas usando o encadeamento de perfis do IAM.

Melhorias nesta versão do mecanismo

- Melhoria da etapa `fail()` do Gremlin para diferenciar a exceção produzida de uma `InternalFailureException` genérica e garantir que qualquer mensagem fornecida pelo usuário fosse propagada de volta ao chamador.
- Melhoria das otimizações do mecanismo de consulta do Gremlin para `store`, `aggregate`, `cap`, `limit` e `hasLabel`.

- Adição de suporte para funções trignométricas do openCypher:
  - `acos()`
  - `asin()`
  - `atan()`
  - `atan2()`
  - `cos()`
  - `cot()`
  - `degrees()`
  - `pi()`
  - `radians()`
  - `sin()`
  - `tan()`
- Adição de suporte para várias funções de agregação do openCypher:
  - `percentileDisc()`
  - `stDev()`
- Adição de suporte para a função `epochmillis()` do openCypher que converte um `datetime` em `epochmillis`. Por exemplo:

```
MATCH (n) RETURN epochMillis(n.someDateTime)
1698972364782
```

- Adição de suporte para o operador (`%`) do módulo do openCypher.
- Adição de suporte para a ferramenta openCypher Static Debug Explain.
- Adição de suporte para a função `randomUUID()` no openCypher.
- Melhoria do desempenho do openCypher:
  - Melhoria do analisador e do planejador de consultas.
  - Melhoria da utilização da CPU no mecanismo do DFE.
  - Melhoria no desempenho de consultas que contêm várias cláusulas de atualização que reutilizam as mesmas variáveis. Veja estes exemplos:

```
MERGE (n {name: 'John'})
  or
MERGE (m {name: 'Jim'})
```

```
  or
MERGE (n)-[:knows {since: 2023}]#(m)
```

- Otimização dos planos de consulta para padrões de consulta de vários saltos, como:

```
MATCH (n)-->()-->()-->(m)
RETURN n m
```

- Melhoria no desempenho da injeção de listas e mapas por meio de consultas parametrizadas. Por exemplo:

```
UNWIND $idList as id MATCH (n {`~id`: id})
RETURN n.name
```

- Melhoria na execução de consultas que contêm `WITH` tornando-as uma barreira apropriada.
- Otimização para evitar a materialização redundante de valores em `Unfold` e funções de agregação.

- Melhoria no desempenho das consultas do SPARQL que contêm um grande número de entradas estáticas na cláusula `VALUES`, como:

```
SELECT ?n WHERE { VALUES (?name) { ("John") ("Jim") ... many values ... } ?n a ?
n_type . ?n ?name . }
```

- Melhoria no desempenho de consultas SPARQL CBD.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro do Gremlin em que consultas longas com aninhamento profundo causavam alto uso da CPU e tempos limite de consulta durante a fase de planejamento da consulta.
- Correção de um erro do Gremlin em que uma `NullPointerException` inválida podia ser lançada ao usar `mergeV` ou `mergeE`.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.2.1.0.R3, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Versão compatível mais antiga do Gremlin: `3.6.2`

- Versão compatível mais recente do Gremlin: `3.6.2`
- openCypher versão: `Neptune-9.0.20190305-1.0`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.2.1.0.R3 do mecanismo

Atualizar para esta versão

O Amazon Neptune 1.2.1.0.R3 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.2.1.0.R2 (02/05/2023)

Desde 02/05/2023, a versão 1.2.1.0.R2 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

> **Note**
>
> Se estiver fazendo a atualização de uma versão do mecanismo anterior à 1.2.0.0:
>
> - A versão 1.2.0.0 do mecanismo introduziu um novo formato para grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados. Como resultado, se você estiver atualizando de uma versão de mecanismo anterior à 1.2.0.0 para a versão 1.2.0.0 ou posterior, deverá recriar todos os grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados existentes usando a família de grupos de parâmetros `neptune1.2`. As versões anteriores usavam a família de grupos de parâmetros `neptune1`, e esses grupos de parâmetros não funcionarão com a versão 1.2.0.0 e posterior. Consulte Grupos de parâmetros do Amazon Neptune para obter mais informações.
> - A versão 1.2.0.0 do mecanismo também introduziu um novo formato para undo logs. Como resultado, todos os undo logs criados por uma versão anterior do mecanismo devem ser eliminados e a métrica `UndoLogsListSize` do CloudWatch deve cair para zero para que seja possível iniciar qualquer atualização de uma versão anterior à 1.2.0.0. Se houver muitos registros de undo logs (duzentos mil ou mais) ao tentar iniciar uma atualização, a tentativa de atualização poderá expirar enquanto aguarda a conclusão da limpeza dos undo logs.
>
>   É possível acelerar a taxa de limpeza atualizando a instância de gravador do cluster, que é onde a limpeza ocorre. Fazer isso antes de tentar realizar a atualização pode reduzir o número de undo logs antes de começar. Aumentar o tamanho do gravador para um tipo de instância 24XL pode aumentar a taxa de limpeza para mais de um milhão de registros por hora.
>
>   Se a métrica `UndoLogsListSize` do CloudWatch for extremamente grande, abrir um caso de suporte pode ajudar a examinar estratégias adicionais para reduzi-la.

- Por fim, houve uma alteração significativa na versão 1.2.0.0. Ela afeta o código anterior que usava o protocolo Bolt com autenticação do IAM. A partir da versão 1.2.0.0, o Bolt precisa de um caminho de recursos para a assinatura do IAM. Em Java, a definição do caminho de recursos pode ser assim: `request.setResourcePath("/openCypher"));`. Em outras linguagens, o `/openCypher` pode ser anexado ao URI do endpoint. Consulte Usar o protocolo Bolt para ver exemplos.

Melhorias nesta versão do mecanismo

- Adição de um parâmetro `enableInterContainerTrafficEncryption` a todas as APIs do Neptune ML, que você pode usar para habilitar e desabilitar a criptografia de tráfego entre contêineres em trabalhos de treinamento ou ajuste de hiperparâmetros.
- Adição de suporte a vários rótulos para `mergeV()` e `mergeE()` do Gremlin.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro do openCypher em que as consultas de atualização e retorno não tratavam `orderBy`, `limit` e `skip` corretamente.
- Correção de um erro do openCypher que permitia que parâmetros contidos em uma solicitação fossem substituídos por parâmetros contidos em outra solicitação simultânea.
- Correção de um erro do openCypher em que os logs de consulta lenta não continham os tempos de consulta corretos.
- Correção de um erro do Gremlin em que um vazamento de transação podia ocorrer quando uma consulta contendo `GroupCountStep` era enviada como uma string.
- Correção de um erro do Gremlin em que as consultas do WebSocket falhavam quando os logs de consultas lentas estavam habilitados.
- Correção de um erro do Gremlin em que os logs de depuração do contador de armazenamento ficavam ausentes dos logs de consultas lentas para solicitações do WebSocket.
- Correção de alguns erros do Gremlin que envolviam `mergeV()` e `mergeE()`.
- Correção de um erro do SPARQL em que os custos de consultas de grafos nomeados eram estimados incorretamente, resultando em planos de consulta abaixo do ideal e erros de falta de memória.

- Correção de um erro que afetava a autorização para consultas do Gremlin e do openCypher em um cluster habilitado para IAM.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.2.1.0.R2, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Versão compatível mais antiga do Gremlin: `3.6.2`
- Versão compatível mais recente do Gremlin: `3.6.2`
- openCypher versão: `Neptune-9.0.20190305-1.0`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.2.1.0.R2 do mecanismo

Atualizar para esta versão

O Amazon Neptune 1.2.1.0.R2 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
> Cannot modify engine version because instance (instance identifier) is
> ```

```
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.2.0.2 (20/11/2022)

Desde 20/11/2022, a versão 1.2.0.2 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Note

Se estiver fazendo a atualização de uma versão do mecanismo anterior à 1.2.0.0:

- A versão 1.2.0.0 do mecanismo introduziu um novo formato para grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados. Como resultado, se você estiver atualizando de uma versão de mecanismo anterior à 1.2.0.0 para a versão 1.2.0.0 ou posterior, deverá recriar todos os grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados existentes usando a família de grupos de parâmetros `neptune1.2`. As versões anteriores usavam a família de grupos de parâmetros `neptune1`, e esses grupos de parâmetros não funcionarão com a versão 1.2.0.0 e posterior. Consulte Grupos de parâmetros do Amazon Neptune para obter mais informações.
- A versão 1.2.0.0 do mecanismo também introduziu um novo formato para undo logs. Como resultado, todos os undo logs criados por uma versão anterior do mecanismo devem ser eliminados e a métrica `UndoLogsListSize` do CloudWatch deve cair para zero para que seja possível iniciar qualquer atualização de uma versão anterior à 1.2.0.0. Se houver muitos registros de undo logs (duzentos mil ou mais) ao tentar iniciar uma atualização, a tentativa de atualização poderá expirar enquanto aguarda a conclusão da limpeza dos undo logs.

É possível acelerar a taxa de limpeza atualizando a instância de gravador do cluster, que é onde a limpeza ocorre. Fazer isso antes de tentar realizar a atualização pode reduzir o número de undo logs antes de começar. Aumentar o tamanho do gravador para um tipo de instância 24XL pode aumentar a taxa de limpeza para mais de um milhão de registros por hora.

Se a métrica `UndoLogsListSize` do CloudWatch for extremamente grande, abrir um caso de suporte pode ajudar a examinar estratégias adicionais para reduzi-la.

- Por fim, houve uma alteração significativa na versão 1.2.0.0. Ela afeta o código anterior que usava o protocolo Bolt com autenticação do IAM. A partir da versão 1.2.0.0, o Bolt precisa de um caminho de recursos para a assinatura do IAM. Em Java, a definição do caminho de recursos pode ser assim: `request.setResourcePath("/openCypher"));`. Em outras linguagens, o `/openCypher` pode ser anexado ao URI do endpoint. Consulte Usar o protocolo Bolt para ver exemplos.

Versões de patch subsequentes para esta versão

- Versão: 1.2.0.2.R2 (15/12/2022)
- Versão: 1.2.0.2.R3 (27/03/2023)
- Versão: 1.2.0.2.R4 (08/05/2023)
- Versão: 1.2.0.2.R5 (16/08/2023)
- Versão: 1.2.0.2.R6 (12/09/2023)

Novos recursos nesta versão do mecanismo

- Introdução da inferência indutiva em tempo real para Gremlin no Neptune ML.
- Introdução de uma extensão do openCypher compatível com a especificação de valores de ID personalizados para entidades em vez dos UUIDs gerados pelo Neptune. A capacidade de atribuir IDs personalizados facilita a migração do Neo4j para o Neptune.

> **Warning**
>
> Essa extensão da especificação do openCypher é incompatível com versões anteriores, porque `~id` agora é considerado um nome de propriedade reservado. Se você já estiver

usando `~id` como propriedade nos dados e nas consultas, deverá migrar a propriedade `~id` para uma nova chave de propriedade antes de realizar a atualização para esta versão.

- Adição de vários novos modos `DESCRIBE` do SPARQL junto com dicas de consulta para configurá-los.

Melhorias nesta versão do mecanismo

- Melhoria no desempenho do openCypher, especialmente para consultas VLP.
- Melhoria no desempenho do DFE para consultas do Gremlin com limites não terminais, como:

```
g.withSideEffect('Neptune#useDFE',true).V().hasLabel('Student').limit(5).out('takesCourse')
```

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.2.0.2, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Versão compatível mais antiga do Gremlin: `3.5.2`
- Versão compatível mais recente do Gremlin: `3.5.4`
- openCypher versão: `Neptune-9.0.20190305-1.0`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.2.0.2 do mecanismo

Atualizar para esta versão

O Amazon Neptune 1.2.0.2 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot

manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.2.0.2.R6 (12/09/2023)

Desde 12/09/2023, a versão 1.2.0.2.R6 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

> **Note**
>
> Se estiver fazendo a atualização de uma versão do mecanismo anterior à 1.2.0.0:
>
> - A versão 1.2.0.0 do mecanismo introduziu um novo formato para grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados. Como resultado, se você estiver atualizando de uma versão de mecanismo anterior à 1.2.0.0 para a versão 1.2.0.0 ou posterior, deverá recriar todos os grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados existentes usando a família de grupos

de parâmetros `neptune1.2`. As versões anteriores usavam a família de grupos de parâmetros `neptune1`, e esses grupos de parâmetros não funcionarão com a versão 1.2.0.0 e posterior. Consulte Grupos de parâmetros do Amazon Neptune para obter mais informações.

- A versão 1.2.0.0 do mecanismo também introduziu um novo formato para undo logs. Como resultado, todos os undo logs criados por uma versão anterior do mecanismo devem ser eliminados e a métrica `UndoLogsListSize` do CloudWatch deve cair para zero para que seja possível iniciar qualquer atualização de uma versão anterior à 1.2.0.0. Se houver muitos registros de undo logs (duzentos mil ou mais) ao tentar iniciar uma atualização, a tentativa de atualização poderá expirar enquanto aguarda a conclusão da limpeza dos undo logs.

  É possível acelerar a taxa de limpeza atualizando a instância de gravador do cluster, que é onde a limpeza ocorre. Fazer isso antes de tentar realizar a atualização pode reduzir o número de undo logs antes de começar. Aumentar o tamanho do gravador para um tipo de instância 24XL pode aumentar a taxa de limpeza para mais de um milhão de registros por hora.

  Se a métrica `UndoLogsListSize` do CloudWatch for extremamente grande, abrir um caso de suporte pode ajudar a examinar estratégias adicionais para reduzi-la.
- Por fim, houve uma alteração significativa na versão 1.2.0.0. Ela afeta o código anterior que usava o protocolo Bolt com autenticação do IAM. A partir da versão 1.2.0.0, o Bolt precisa de um caminho de recursos para a assinatura do IAM. Em Java, a definição do caminho de recursos pode ser assim: `request.setResourcePath("/openCypher"));`. Em outras linguagens, o `/openCypher` pode ser anexado ao URI do endpoint. Consulte Usar o protocolo Bolt para ver exemplos.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro do SPARQL em que o operador `REGEX` nunca tinha êxito quando chamado em um literal marcado com uma linguagem.
- Correção de um problema que causava degradação do desempenho do carregamento em massa.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.2.0.2.R6, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Versão compatível mais antiga do Gremlin: `3.5.2`
- Versão compatível mais recente do Gremlin: `3.5.5`
- openCypher versão: `Neptune-9.0.20190305-1.0`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.2.0.2.R6 do mecanismo

O cluster de banco de dados do Neptune será atualizado com esta versão de patch de manutenção automaticamente durante a janela de manutenção seguinte se você estiver executando a versão `1.2.0.2` do mecanismo.

Atualizar para esta versão

O Amazon Neptune 1.2.0.2.R6 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
> ```

```
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.2.0.2.R5 (16/08/2023)

Desde 16/08/2023, a versão 1.2.0.2.R5 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Important

As alterações introduzidas nesta versão do mecanismo, em alguns casos, podem ocasionar degradação do desempenho do carregamento em massa. Como resultado, as atualizações para esta versão foram temporariamente suspensas até que o problema seja resolvido.

Note

Se estiver fazendo a atualização de uma versão do mecanismo anterior à 1.2.0.0:

- A versão 1.2.0.0 do mecanismo introduziu um novo formato para grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados. Como resultado, se você estiver atualizando de uma versão de mecanismo anterior à 1.2.0.0 para a versão 1.2.0.0 ou posterior, deverá recriar todos os grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados existentes usando a família de grupos de parâmetros `neptune1.2`. As versões anteriores usavam a família de grupos de parâmetros `neptune1`, e esses grupos de parâmetros não funcionarão com a versão 1.2.0.0 e posterior. Consulte Grupos de parâmetros do Amazon Neptune para obter mais informações.

- A versão 1.2.0.0 do mecanismo também introduziu um novo formato para undo logs. Como resultado, todos os undo logs criados por uma versão anterior do mecanismo devem ser eliminados e a métrica `UndoLogsListSize` do CloudWatch deve cair para zero para que seja possível iniciar qualquer atualização de uma versão anterior à 1.2.0.0. Se houver muitos registros de undo logs (duzentos mil ou mais) ao tentar iniciar uma atualização, a tentativa de atualização poderá expirar enquanto aguarda a conclusão da limpeza dos undo logs.

  É possível acelerar a taxa de limpeza atualizando a instância de gravador do cluster, que é onde a limpeza ocorre. Fazer isso antes de tentar realizar a atualização pode reduzir o número de undo logs antes de começar. Aumentar o tamanho do gravador para um tipo de instância 24XL pode aumentar a taxa de limpeza para mais de um milhão de registros por hora.

  Se a métrica `UndoLogsListSize` do CloudWatch for extremamente grande, abrir um caso de suporte pode ajudar a examinar estratégias adicionais para reduzi-la.
- Por fim, houve uma alteração significativa na versão 1.2.0.0. Ela afeta o código anterior que usava o protocolo Bolt com autenticação do IAM. A partir da versão 1.2.0.0, o Bolt precisa de um caminho de recursos para a assinatura do IAM. Em Java, a definição do caminho de recursos pode ser assim: `request.setResourcePath("/openCypher"));`. Em outras linguagens, o `/openCypher` pode ser anexado ao URI do endpoint. Consulte Usar o protocolo Bolt para ver exemplos.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro do Gremlin em que `order()` não classificava corretamente as saídas de string quando algumas delas continham um caractere de espaço.
- Correção de um erro do Gremlin em que ocorria um vazamento de transação ao conferir o endpoint de status da consulta do Gremlin para consultas com predicados em percursos secundários para etapas não processadas de modo nativo.
- Correção de um erro do openCypher no tratamento de transações do Bolt.
- Correção de um problema de simultaneidade na camada de armazenamento que poderia causar uma falha.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.2.0.2.R5, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Versão compatível mais antiga do Gremlin: `3.5.2`
- Versão compatível mais recente do Gremlin: `3.5.5`
- openCypher versão: `Neptune-9.0.20190305-1.0`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.2.0.2.R5 do mecanismo

O cluster de banco de dados do Neptune será atualizado com esta versão de patch de manutenção automaticamente durante a janela de manutenção seguinte se você estiver executando a versão `1.2.0.2` do mecanismo.

Atualizar para esta versão

O Amazon Neptune 1.2.0.2.R5 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
> ```

```
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.2.0.2.R4 (08/05/2023)

Desde 08/05/2023, a versão 1.2.0.2.R4 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Note

Se estiver fazendo a atualização de uma versão do mecanismo anterior à 1.2.0.0:

- A versão 1.2.0.0 do mecanismo introduziu um novo formato para grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados. Como resultado, se você estiver atualizando de uma versão de mecanismo anterior à 1.2.0.0 para a versão 1.2.0.0 ou posterior, deverá recriar todos os grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados existentes usando a família de grupos de parâmetros `neptune1.2`. As versões anteriores usavam a família de grupos de parâmetros `neptune1`, e esses grupos de parâmetros não funcionarão com a versão 1.2.0.0 e posterior. Consulte Grupos de parâmetros do Amazon Neptune para obter mais informações.
- A versão 1.2.0.0 do mecanismo também introduziu um novo formato para undo logs. Como resultado, todos os undo logs criados por uma versão anterior do mecanismo devem ser eliminados e a métrica `UndoLogsListSize` do CloudWatch deve cair para zero para que seja possível iniciar qualquer atualização de uma versão anterior à 1.2.0.0. Se houver muitos registros de undo logs (duzentos mil ou mais) ao tentar iniciar uma atualização, a tentativa de atualização poderá expirar enquanto aguarda a conclusão da limpeza dos undo logs.

É possível acelerar a taxa de limpeza atualizando a instância de gravador do cluster, que é onde a limpeza ocorre. Fazer isso antes de tentar realizar a atualização pode reduzir o número de undo logs antes de começar. Aumentar o tamanho do gravador para um tipo de instância 24XL pode aumentar a taxa de limpeza para mais de um milhão de registros por hora.

Se a métrica `UndoLogsListSize` do CloudWatch for extremamente grande, abrir um caso de suporte pode ajudar a examinar estratégias adicionais para reduzi-la.

- Por fim, houve uma alteração significativa na versão 1.2.0.0. Ela afeta o código anterior que usava o protocolo Bolt com autenticação do IAM. A partir da versão 1.2.0.0, o Bolt precisa de um caminho de recursos para a assinatura do IAM. Em Java, a definição do caminho de recursos pode ser assim: `request.setResourcePath("/openCypher"));`. Em outras linguagens, o `/openCypher` pode ser anexado ao URI do endpoint. Consulte Usar o protocolo Bolt para ver exemplos.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro do SPARQL em que um grande número de valores injetados por uma cláusula `VALUES` poderia causar degradação do desempenho.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.2.0.2.R4, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Versão compatível mais antiga do Gremlin: `3.5.2`
- Versão compatível mais recente do Gremlin: `3.5.6`
- openCypher versão: `Neptune-9.0.20190305-1.0`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.2.0.2.R4 do mecanismo

O cluster de banco de dados do Neptune será atualizado com esta versão de patch de manutenção automaticamente durante a janela de manutenção seguinte se você estiver executando a versão `1.2.0.2` do mecanismo.

Atualizar para esta versão

O Amazon Neptune 1.2.0.2.R4 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para essa versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.2.0.2.R3 (27/03/2023)

Desde 27/03/2023, a versão 1.2.0.2.R3 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Note

Se estiver fazendo a atualização de uma versão do mecanismo anterior à 1.2.0.0:

- A versão 1.2.0.0 do mecanismo introduziu um novo formato para grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados. Como resultado, se você estiver atualizando de uma versão de mecanismo anterior à 1.2.0.0 para a versão 1.2.0.0 ou posterior, deverá recriar todos os grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados existentes usando a família de grupos de parâmetros `neptune1.2`. As versões anteriores usavam a família de grupos de parâmetros `neptune1`, e esses grupos de parâmetros não funcionarão com a versão 1.2.0.0 e posterior. Consulte Grupos de parâmetros do Amazon Neptune para obter mais informações.
- A versão 1.2.0.0 do mecanismo também introduziu um novo formato para undo logs. Como resultado, todos os undo logs criados por uma versão anterior do mecanismo devem ser eliminados e a métrica `UndoLogsListSize` do CloudWatch deve cair para zero para que seja possível iniciar qualquer atualização de uma versão anterior à 1.2.0.0. Se houver muitos registros de undo logs (duzentos mil ou mais) ao tentar iniciar uma atualização, a tentativa de atualização poderá expirar enquanto aguarda a conclusão da limpeza dos undo logs.

  É possível acelerar a taxa de limpeza atualizando a instância de gravador do cluster, que é onde a limpeza ocorre. Fazer isso antes de tentar realizar a atualização pode reduzir o número de undo logs antes de começar. Aumentar o tamanho do gravador para um tipo de instância 24XL pode aumentar a taxa de limpeza para mais de um milhão de registros por hora.

  Se a métrica `UndoLogsListSize` do CloudWatch for extremamente grande, abrir um caso de suporte pode ajudar a examinar estratégias adicionais para reduzi-la.
- Por fim, houve uma alteração significativa na versão 1.2.0.0. Ela afeta o código anterior que usava o protocolo Bolt com autenticação do IAM. A partir da versão 1.2.0.0, o Bolt precisa de um caminho de recursos para a assinatura do IAM. Em Java, a definição do caminho de recursos pode ser assim: `request.setResourcePath("/openCypher"));`. Em outras linguagens, o `/openCypher` pode ser anexado ao URI do endpoint. Consulte Usar o protocolo Bolt para ver exemplos.

Melhorias nesta versão do mecanismo

- Para clusters de banco de dados sem servidor, foram alteradas a configuração de capacidade mínima para 1,0 NCU e a configuração máxima válida mais baixa para 2,5 NCUs. Consulte Escalabilidade da capacidade em um cluster de banco de dados do Neptune Serverless
- Adição de um parâmetro `enableInterContainerTrafficEncryption` a todas as APIs do Neptune ML, que você pode usar para habilitar e desabilitar a criptografia de tráfego entre contêineres em trabalhos de treinamento ou ajuste de hiperparâmetros.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro do Gremlin em que `option(Predicate)` não estava sendo reconhecida como uma sintaxe válida do Gremlin.
- Correção de um erro do Gremlin que fazia com que as consultas não fossem limpas adequadamente se falhassem porque continham muitas etapas.
- Correção de um problema de exatidão do Gremlin que afetava consultas do DFE com `limit` como percurso filho de etapas que não são de junção retornando ao Tinkerpop. Veja um exemplo de consulta desse tipo:

```
g.withSideEffect('Neptune#useDFE', true).V().as("a").select("a").by(out().limit(1))
```

- Correção de um possível vazamento de transação do Gremlin quando uma consulta enviada como uma string contém `GroupCountStep`.
- Correção de um erro do openCypher em que o tipo de valor de parâmetro não era inferido em uma lista ou uma lista de mapas.
- Correção de um erro do openCypher em que as consultas de atualização e retorno não tratavam `orderBy`, `limit` e `skip` corretamente.
- Correção de um erro do openCypher que permitia que parâmetros contidos em uma solicitação fossem substituídos por parâmetros contidos em outra solicitação simultânea.
- Correção de um erro do SPARQL em que um grande número de valores injetados em uma cláusula `VALUES` poderia causar degradação do desempenho.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.2.0.2.R3, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Versão compatível mais antiga do Gremlin: `3.5.2`
- Versão compatível mais recente do Gremlin: `3.5.6`
- openCypher versão: `Neptune-9.0.20190305-1.0`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.2.0.2.R3 do mecanismo

O cluster de banco de dados do Neptune será atualizado com esta versão de patch de manutenção automaticamente durante a janela de manutenção seguinte se você estiver executando a versão `1.2.0.2` do mecanismo.

Atualizar para esta versão

O Amazon Neptune 1.2.0.2.R3 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
> ```

```
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.2.0.2.R2 (15/12/2022)

Desde 15/12/2022, a versão 1.2.0.2.R2 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Note

Se estiver fazendo a atualização de uma versão do mecanismo anterior à 1.2.0.0:

- A versão 1.2.0.0 do mecanismo introduziu um novo formato para grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados. Como resultado, se você estiver atualizando de uma versão de mecanismo anterior à 1.2.0.0 para a versão 1.2.0.0 ou posterior, deverá recriar todos os grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados existentes usando a família de grupos de parâmetros `neptune1.2`. As versões anteriores usavam a família de grupos de parâmetros `neptune1`, e esses grupos de parâmetros não funcionarão com a versão 1.2.0.0 e posterior. Consulte Grupos de parâmetros do Amazon Neptune para obter mais informações.
- A versão 1.2.0.0 do mecanismo também introduziu um novo formato para undo logs. Como resultado, todos os undo logs criados por uma versão anterior do mecanismo devem ser eliminados e a métrica `UndoLogsListSize` do CloudWatch deve cair para zero para que seja possível iniciar qualquer atualização de uma versão anterior à 1.2.0.0. Se houver muitos registros de undo logs (duzentos mil ou mais) ao tentar iniciar uma atualização, a tentativa de atualização poderá expirar enquanto aguarda a conclusão da limpeza dos undo logs.

É possível acelerar a taxa de limpeza atualizando a instância de gravador do cluster, que é onde a limpeza ocorre. Fazer isso antes de tentar realizar a atualização pode reduzir o número de undo logs antes de começar. Aumentar o tamanho do gravador para um tipo de instância 24XL pode aumentar a taxa de limpeza para mais de um milhão de registros por hora.

Se a métrica `UndoLogsListSize` do CloudWatch for extremamente grande, abrir um caso de suporte pode ajudar a examinar estratégias adicionais para reduzi-la.

- Por fim, houve uma alteração significativa na versão 1.2.0.0. Ela afeta o código anterior que usava o protocolo Bolt com autenticação do IAM. A partir da versão 1.2.0.0, o Bolt precisa de um caminho de recursos para a assinatura do IAM. Em Java, a definição do caminho de recursos pode ser assim: `request.setResourcePath("/openCypher"));`. Em outras linguagens, o `/openCypher` pode ser anexado ao URI do endpoint. Consulte Usar o protocolo Bolt para ver exemplos.

Melhorias nesta versão do mecanismo

- Melhoria no desempenho das consultas do openCypher que envolvem `MERGE` e `OPTIONAL MATCH`.
- Melhoria no desempenho das consultas do openCypher que envolvem `UNWIND` de uma lista de mapas de valores literais.
- Melhoria no desempenho das consultas do openCypher que têm um filtro `IN` para `id`. Por exemplo:

```
MATCH (n) WHERE id(n) IN ['1', '2', '3'] RETURN n
```

- Melhorias no desempenho e correções para vários operadores do Gremlin, incluindo `repeat`, `coalesce`, `store` e `aggregate`.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro do openCypher em que as consultas geravam uma string, `"null"`, em vez de um valor nulo em Bolt e SPARQL-JSON.
- Correção de um erro do Gremlin que fazia com que um rótulo de etapa anexado a `UnionStep` não fosse propagado para o último elemento do caminho dos percursos secundários.

- Correção de um erro do Gremlin que fazia com que `valueMap()` não fosse otimizado em um percurso `by()` no mecanismo do DFE.
- Correção de um erro do Gremlin em que consultas de leitura executadas como parte de uma transação mais longa do Gremlin não bloqueavam as linhas.
- Correção de um erro no log de auditoria que fazia com que informações desnecessárias fossem registradas e determinados campos ficassem ausentes nos logs.
- Correção de um erro no log de auditoria em que o ARN do IAM das solicitações HTTP para um cluster de banco de dados habilitado para IAM não era registrado.
- Correção de um erro no cache de pesquisa para limitar a memória incremental usada para gravações no cache.
- Correção de um erro do cache de pesquisa que envolvia a configuração do modo somente leitura para o cache de pesquisa quando as gravações falhavam.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.2.0.2.R2, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Versão compatível mais antiga do Gremlin: `3.5.2`
- Versão compatível mais recente do Gremlin: `3.5.4`
- openCypher versão: `Neptune-9.0.20190305-1.0`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.2.0.2.R2 do mecanismo

O cluster de banco de dados do Neptune será atualizado com esta versão de patch de manutenção automaticamente durante a janela de manutenção seguinte se você estiver executando a versão `1.2.0.2` do mecanismo.

Atualizar para esta versão

O Amazon Neptune 1.2.0.2.R2 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora.

Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.2.0.1 (26/10/2022)

Desde 26/10/2022, a versão 1.2.0.1 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

> **Note**
>
> Se estiver fazendo a atualização de uma versão do mecanismo anterior à 1.2.0.0:

- A versão 1.2.0.0 do mecanismo introduziu um novo formato para grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados. Como resultado, se você estiver atualizando de uma versão de mecanismo anterior à 1.2.0.0 para a versão 1.2.0.0 ou posterior, deverá recriar todos os grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados existentes usando a família de grupos de parâmetros `neptune1.2`. As versões anteriores usavam a família de grupos de parâmetros `neptune1`, e esses grupos de parâmetros não funcionarão com a versão 1.2.0.0 e posterior. Consulte Grupos de parâmetros do Amazon Neptune para obter mais informações.
- A versão 1.2.0.0 do mecanismo também introduziu um novo formato para undo logs. Como resultado, todos os undo logs criados por uma versão anterior do mecanismo devem ser eliminados e a métrica `UndoLogsListSize` do CloudWatch deve cair para zero para que seja possível iniciar qualquer atualização de uma versão anterior à 1.2.0.0. Se houver muitos registros de undo logs (duzentos mil ou mais) ao tentar iniciar uma atualização, a tentativa de atualização poderá expirar enquanto aguarda a conclusão da limpeza dos undo logs.

  É possível acelerar a taxa de limpeza atualizando a instância de gravador do cluster, que é onde a limpeza ocorre. Fazer isso antes de tentar realizar a atualização pode reduzir o número de undo logs antes de começar. Aumentar o tamanho do gravador para um tipo de instância 24XL pode aumentar a taxa de limpeza para mais de um milhão de registros por hora.

  Se a métrica `UndoLogsListSize` do CloudWatch for extremamente grande, abrir um caso de suporte pode ajudar a examinar estratégias adicionais para reduzi-la.
- Por fim, houve uma alteração significativa na versão 1.2.0.0. Ela afeta o código anterior que usava o protocolo Bolt com autenticação do IAM. A partir da versão 1.2.0.0, o Bolt precisa de um caminho de recursos para a assinatura do IAM. Em Java, a definição do caminho de recursos pode ser assim: `request.setResourcePath("/openCypher"));`. Em outras linguagens, o `/openCypher` pode ser anexado ao URI do endpoint. Consulte Usar o protocolo Bolt para ver exemplos.

Versões de patch subsequentes para esta versão

- Versão de manutenção: 1.2.0.1.R2 (13/12/2022)

- Versão de manutenção: 1.2.0.1.R3 (27/09/2023)

Novos recursos nesta versão do mecanismo

- Introdução do Amazon Neptune Serverless, uma configuração de ajuste de escala automático sob demanda que aumenta a escala do cluster de banco de dados verticalmente para atender aos aumentos na demanda de processamento e depois torna a reduz a escala verticalmente quando a demanda diminui.

Melhorias nesta versão do mecanismo

- Melhoria no desempenho de consultas `order-by` do Gremlin. As consultas do Gremlin com um `order-by` no final de uma `NeptuneGraphQueryStep` agora usam um tamanho de bloco maior para melhor desempenho. Isso não se aplica a `order-by` em um nó interno (não raiz) do plano de consulta.
- Melhoria do desempenho de consultas de atualização do Gremlin. Agora, vértices e bordas devem ser bloqueados contra exclusão ao adicionar bordas ou propriedades. Essa alteração elimina bloqueios duplicados em uma transação, o que melhora o desempenho.
- Melhoria no desempenho das consultas do Gremlin que usam `dedup()` em uma subconsulta `repeat()` enviando `dedup` à camada de execução nativa.
- Adição de mensagens de erro fáceis de usar para erros de autenticação do IAM. Agora, essas mensagens mostram o ARN do usuário ou do perfil do IAM, o ARN do recurso e uma lista de ações não autorizadas da solicitação. A lista de ações não autorizadas ajuda você a ver o que pode estar faltando ou é explicitamente negado na política do IAM que você está usando.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro do Gremlin em que o uso de `PartitionStrategy` após a atualização para o TinkerPop 3.5 gerava incorretamente um erro com a mensagem “PartitionStrategy não funciona com percursos anônimos”, o que impedia a execução do percurso.
- Correção de um erro de exatidão do Gremlin envolvendo a conversão de `WherePredicateStep`, em que o mecanismo de consulta do Neptune estava produzindo resultados incorretos para consultas que usavam `where(P.neq('x'))` e as variações dela.

- Correção de um erro do openCypher na cláusula `MERGE` que, em alguns casos, causava a criação duplicada de nós e bordas.
- Correção de um erro no tratamento de consultas do SPARQL que contêm `(NOT) EXISTS` em uma cláusula `OPTIONAL` em que, em alguns casos, os resultados da consulta ficavam ausentes.
- Correção de um erro do carregador em massa que causava regressões de desempenho sob cargas de inserção pesadas.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.2.0.1, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Versão compatível mais antiga do Gremlin: `3.5.2`
- Versão compatível mais recente do Gremlin: `3.5.4`
- openCypher versão: `Neptune-9.0.20190305-1.0`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.2.0.1 do mecanismo

Atualizar para esta versão

O Amazon Neptune 1.2.0.1 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.1 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.1 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.2.0.1.R3 (27/09/2023)

Desde 27/09/2023, a versão 1.2.0.1.R3 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

> **Note**
>
> Se estiver fazendo a atualização de uma versão do mecanismo anterior à 1.2.0.0:
>
> - A versão 1.2.0.0 do mecanismo introduziu um novo formato para grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados. Como resultado, se você estiver atualizando de uma versão de mecanismo anterior à 1.2.0.0 para a versão 1.2.0.0 ou posterior, deverá recriar todos os grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados existentes usando a família de grupos de parâmetros `neptune1.2`. As versões anteriores usavam a família de grupos de parâmetros `neptune1`, e esses grupos de parâmetros não funcionarão com a versão 1.2.0.0 e posterior. Consulte Grupos de parâmetros do Amazon Neptune para obter mais informações.

- A versão 1.2.0.0 do mecanismo também introduziu um novo formato para undo logs. Como resultado, todos os undo logs criados por uma versão anterior do mecanismo devem ser eliminados e a métrica `UndoLogsListSize` do CloudWatch deve cair para zero para que seja possível iniciar qualquer atualização de uma versão anterior à 1.2.0.0. Se houver muitos registros de undo logs (duzentos mil ou mais) ao tentar iniciar uma atualização, a tentativa de atualização poderá expirar enquanto aguarda a conclusão da limpeza dos undo logs.

  É possível acelerar a taxa de limpeza atualizando a instância de gravador do cluster, que é onde a limpeza ocorre. Fazer isso antes de tentar realizar a atualização pode reduzir o número de undo logs antes de começar. Aumentar o tamanho do gravador para um tipo de instância 24XL pode aumentar a taxa de limpeza para mais de um milhão de registros por hora.

  Se a métrica `UndoLogsListSize` do CloudWatch for extremamente grande, abrir um caso de suporte pode ajudar a examinar estratégias adicionais para reduzi-la.
- Por fim, houve uma alteração significativa na versão 1.2.0.0. Ela afeta o código anterior que usava o protocolo Bolt com autenticação do IAM. A partir da versão 1.2.0.0, o Bolt precisa de um caminho de recursos para a assinatura do IAM. Em Java, a definição do caminho de recursos pode ser assim: `request.setResourcePath("/openCypher"));`. Em outras linguagens, o `/openCypher` pode ser anexado ao URI do endpoint. Consulte Usar o protocolo Bolt para ver exemplos.

Melhorias nesta versão do mecanismo

- Adição de um parâmetro `enableInterContainerTrafficEncryption` a todas as APIs do Neptune ML, que você pode usar para habilitar e desabilitar a criptografia de tráfego entre contêineres em trabalhos de treinamento ou ajuste de hiperparâmetros.
- Para clusters de banco de dados sem servidor, foram alteradas a configuração de capacidade mínima para 1,0 NCU e a configuração máxima válida mais baixa para 2,5 NCUs. Consulte Escalabilidade da capacidade em um cluster de banco de dados do Neptune Serverless *(antes do lançamento, essa alteração também precisa ser refletida na página sem servidor).*

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro do openCypher em que as consultas de atualização e retorno não tratavam `orderBy`, `limit` e `skip` corretamente.
- Correção de um erro do openCypher que permitia que parâmetros contidos em uma solicitação fossem substituídos por parâmetros contidos em outra solicitação simultânea.
- Correção de um erro do openCypher no tratamento de transações do Bolt.
- Correção dos problemas de exatidão do Gremlin para consultas do DFE com `limit` como percurso filho de etapas que não são de junção retornando ao Tinkerpop. Por exemplo, para consultas como esta:

```
g.withSideEffect('Neptune#useDFE', true)
 .V()
 .as("a")
 .select("a")
 .by(out()
 .limit(1))
```

- Correção de um erro do Gremlin em que uma consulta falhava porque continha muitas etapas do TinkerPop e não era limpa.
- Correção de um erro do Gremlin em que `order()` não classificava corretamente as saídas de string quando algumas delas continham um caractere de espaço.
- Correção de um erro do Gremlin em que um vazamento de transação podia ocorrer quando uma consulta era enviada como uma string e continha `GroupCountStep`.
- Correção de um erro do Gremlin em que ocorria um vazamento de transação ao conferir o endpoint de status da consulta do Gremlin para consultas com predicados em percursos secundários para etapas não processadas de modo nativo.
- Correção de um erro do Gremlin em que adicionar uma borda e suas propriedades seguidas por `inV()` ou `outV()` causava uma `InternalFailureException`.
- Correção de um erro de simultaneidade na camada de armazenamento.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.2.0.1.R3, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Versão compatível mais antiga do Gremlin: `3.5.2`

- Versão compatível mais recente do Gremlin: `3.5.6`
- openCypher versão: `Neptune-9.0.20190305-1.0`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.2.0.1.R3 do mecanismo

O cluster de banco de dados do Neptune será atualizado para esta versão de patch automaticamente durante a janela de manutenção seguinte se você estiver executando a versão 1.2.0.1 do mecanismo.

Atualizar para esta versão

O Amazon Neptune 1.2.0.1.R3 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para essa versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.1 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.1 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```

Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.2.0.1.R2 (13/12/2022)

Desde 13/12/2022, a versão 1.2.0.1.R2 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Note

Se estiver fazendo a atualização de uma versão do mecanismo anterior à 1.2.0.0:

- A versão 1.2.0.0 do mecanismo introduziu um novo formato para grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados. Como resultado, se você estiver atualizando de uma versão de mecanismo anterior à 1.2.0.0 para a versão 1.2.0.0 ou posterior, deverá recriar todos os grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados existentes usando a família de grupos de parâmetros `neptune1.2`. As versões anteriores usavam a família de grupos de parâmetros `neptune1`, e esses grupos de parâmetros não funcionarão com a versão 1.2.0.0 e posterior. Consulte Grupos de parâmetros do Amazon Neptune para obter mais informações.
- A versão 1.2.0.0 do mecanismo também introduziu um novo formato para undo logs. Como resultado, todos os undo logs criados por uma versão anterior do mecanismo devem ser eliminados e a métrica `UndoLogsListSize` do CloudWatch deve cair para zero para que seja possível iniciar qualquer atualização de uma versão anterior à 1.2.0.0. Se houver muitos registros de undo logs (duzentos mil ou mais) ao tentar iniciar uma atualização, a tentativa de atualização poderá expirar enquanto aguarda a conclusão da limpeza dos undo logs.

  É possível acelerar a taxa de limpeza atualizando a instância de gravador do cluster, que é onde a limpeza ocorre. Fazer isso antes de tentar realizar a atualização pode reduzir o

número de undo logs antes de começar. Aumentar o tamanho do gravador para um tipo de instância 24XL pode aumentar a taxa de limpeza para mais de um milhão de registros por hora.

Se a métrica `UndoLogsListSize` do CloudWatch for extremamente grande, abrir um caso de suporte pode ajudar a examinar estratégias adicionais para reduzi-la.

- Por fim, houve uma alteração significativa na versão 1.2.0.0. Ela afeta o código anterior que usava o protocolo Bolt com autenticação do IAM. A partir da versão 1.2.0.0, o Bolt precisa de um caminho de recursos para a assinatura do IAM. Em Java, a definição do caminho de recursos pode ser assim: `request.setResourcePath("/openCypher"));`. Em outras linguagens, o `/openCypher` pode ser anexado ao URI do endpoint. Consulte Usar o protocolo Bolt para ver exemplos.

Melhorias nesta versão do mecanismo

- Melhoria no desempenho das consultas do openCypher que envolvem `UNWIND` em uma lista de mapas de valores literais.
- Melhorias no desempenho e correções para vários operadores do Gremlin, incluindo `repeat`, `coalesce`, `store` e `aggregate`.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro do openCypher em que as consultas geravam uma string, `"null"`, em vez de um valor nulo em Bolt e SPARQL-JSON.
- Correção de um erro no log de auditoria que fazia com que informações desnecessárias fossem registradas e determinados campos ficassem ausentes nos logs.
- Correção de um erro no cache de pesquisa para limitar a memória incremental usada para gravações no cache.
- Correção de um erro do cache de pesquisa que envolvia a configuração do modo somente leitura para o cache de pesquisa quando as gravações falhavam.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.2.0.1.R2, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Versão compatível mais antiga do Gremlin: `3.5.2`
- Versão compatível mais recente do Gremlin: `3.5.4`
- openCypher versão: `Neptune-9.0.20190305-1.0`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.2.0.1.R2 do mecanismo

O cluster de banco de dados do Neptune será atualizado para esta versão de patch automaticamente durante a janela de manutenção seguinte se você estiver executando a versão 1.2.0.1 do mecanismo.

Atualizar para esta versão

O Amazon Neptune 1.2.0.1.R2 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.1 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.1 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```

Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.2.0.0 (21/07/2022)

Desde 21/07/2022, a versão 1.2.0.0 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Note

Se estiver fazendo a atualização de uma versão do mecanismo anterior à 1.2.0.0:

- A versão 1.2.0.0 do mecanismo introduziu um novo formato para grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados. Como resultado, se você estiver atualizando de uma versão de mecanismo anterior à 1.2.0.0 para a versão 1.2.0.0 ou posterior, deverá recriar todos os grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados existentes usando a família de grupos de parâmetros `neptune1.2`. As versões anteriores usavam a família de grupos de parâmetros `neptune1`, e esses grupos de parâmetros não funcionarão com a versão 1.2.0.0 e posterior. Consulte Grupos de parâmetros do Amazon Neptune para obter mais informações.
- A versão 1.2.0.0 do mecanismo também introduziu um novo formato para undo logs. Como resultado, todos os undo logs criados por uma versão anterior do mecanismo devem ser eliminados e a métrica `UndoLogsListSize` do CloudWatch deve cair para zero para que seja possível iniciar qualquer atualização de uma versão anterior à 1.2.0.0. Se houver muitos registros de undo logs (duzentos mil ou mais) ao tentar iniciar uma atualização, a tentativa de atualização poderá expirar enquanto aguarda a conclusão da limpeza dos undo logs.

  É possível acelerar a taxa de limpeza atualizando a instância de gravador do cluster, que é onde a limpeza ocorre. Fazer isso antes de tentar realizar a atualização pode reduzir o

número de undo logs antes de começar. Aumentar o tamanho do gravador para um tipo de instância 24XL pode aumentar a taxa de limpeza para mais de um milhão de registros por hora.

Se a métrica `UndoLogsListSize` do CloudWatch for extremamente grande, abrir um caso de suporte pode ajudar a examinar estratégias adicionais para reduzi-la.

- Por fim, houve uma alteração significativa na versão 1.2.0.0. Ela afeta o código anterior que usava o protocolo Bolt com autenticação do IAM. A partir da versão 1.2.0.0, o Bolt precisa de um caminho de recursos para a assinatura do IAM. Em Java, a definição do caminho de recursos pode ser assim: `request.setResourcePath("/openCypher"));`. Em outras linguagens, o `/openCypher` pode ser anexado ao URI do endpoint. Consulte Usar o protocolo Bolt para ver exemplos.

Versões de patch subsequentes para esta versão

- Versão: 1.2.0.0.R2 (14/10/2022)
- Versão: 1.2.0.0.R3 (15/12/2022)
- Versão: 1.2.0.0.R4 (29/09/2023)

Novos recursos nesta versão do mecanismo

- Adição de suporte para bancos de dados globais. Um banco de dados global do Neptune abrange várias Regiões da AWS e consiste em um cluster de banco de dados principal em uma região e até cinco clusters de banco de dados secundários em outras regiões.
- Adição de suporte para um controle de acesso nas políticas do IAM no Neptune mais granular do que estava disponível anteriormente, com base nas ações do plano de dados. Essa é uma alteração importante, pois as políticas existentes do IAM baseadas na ação `connect` obsoleta devem ser ajustadas para usar as ações mais granulares do plano de dados. Consulte Tipos de política do IAM.
- Melhoria da disponibilidade da instância de leitor. Anteriormente, quando uma instância de gravador era reiniciada, todas as instâncias de leitor em um cluster do Neptune também eram reiniciadas. A partir da versão 1.2.0.0 do mecanismo, as instâncias de leitor permanecem ativas após a reinicialização do gravador, o que melhora a disponibilidade do leitor. As instâncias de leitor podem ser reiniciadas separadamente para captar as alterações do grupo de parâmetros. Consulte Reinicializar uma instância de banco de dados no Amazon Neptune.

- Adição de um novo parâmetro de cluster de banco de dados neptune_streams_expiry_days que permite definir o número de dias em que os registros de fluxos são mantidos no servidor antes de serem excluídos. O intervalo é de um a noventa e o padrão é sete.

Melhorias nesta versão do mecanismo

- Melhoria no desempenho da serialização do Gremlin para consultas ByteCode.
- O Neptune agora processa predicados de texto usando o mecanismo do DFE, para melhorar o desempenho.
- O Neptune agora processa as etapas `limit()` do Gremlin usando o mecanismo do DFE, incluindo limites de percurso não terminal e secundário.
- O tratamento da etapa `union()` do Gremlin pelo DFE foi alterado para funcionar com outros novos atributos, o que significa que os nós de referência aparecem nos perfis de consulta conforme o esperado.
- Melhoria no desempenho em até um fator de cinco de algumas operações de junção caras no DFE por meio da paralelização delas.
- Adição de suporte de modulação `by()` para `OrderGlobalStep order(global)` para o mecanismo do DFE do Gremlin.
- Adição da exibição de valores estáticos injetados nos detalhes de explicação do DFE.
- Melhoria do desempenho ao remover padrões duplicados.
- Adição de suporte à preservação da ordem no mecanismo do DFE do Gremlin.
- Melhoria no desempenho das consultas do Gremlin com filtros vazios, como estes:

```
g.V().hasId(P.within([]))
```

```
g.V().hasId([])
```

- Melhoria das mensagens de erro quando uma consulta do SPARQL usa um valor numérico muito grande para o Neptune representar internamente.
- Melhoria no desempenho para descartar vértices com bordas associadas, reduzindo as pesquisas de índice quando os fluxos estão desabilitados.
- Extensão do suporte do DFE para mais variantes da etapa `has()`, especificamente, para `hasKey()`, `hasLabel()` e para intervalos de predicados para strings/URIs em `has()`. Isso afeta consultas como as seguintes:

```
// hasKey() on properties
g.V().properties().hasKey("name")
g.V().properties().has(T.key, TextP.startingWith("a"))
g.E().properties().hasKey("weight")
g.E().properties().hasKey(TextP.containing("t"))

// hasLabel() on vertex properties
g.V().properties().hasLabel("name")

// range predicates on ID and Label fields
g.V().has(T.label, gt("person"))
g.E().has(T.id, lte("(an ID value)"))
```

- Adição de uma função `join()` do openCypher específica do Neptune que concatena strings em uma lista em uma única string.
- Atualização das políticas gerenciadas do Neptune para incluir permissões de acesso a dados e permissões para as novas APIs de bancos de dados globais.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro em que uma solicitação HTTP sem o tipo de conteúdo especificado falhava automaticamente.
- Correção de um erro do SPARQL no otimizador de consultas que impedia o uso de uma chamada de serviço em uma consulta.
- Correção de um erro do SPARQL no analisador Turtle RDF em que uma combinação específica de dados Unicode causava falha.
- Correção de um erro do SPARQL em que uma combinação específica de cláusulas `GRAPH` e `SELECT` produzia resultados de consulta incorretos.
- Correção de um erro do Gremlin que causava um problema de exatidão em consultas que usavam qualquer etapa de filtro em uma etapa de união, como a seguinte:

```
g.V("1").union(hasLabel("person"), out())
```

- Correção de um erro do Gremlin em que `count()` de `both().simplePath()` resultava no dobro do número real de resultados gerados sem `count()`.

- Correção de um erro do openCypher em que uma exceção de incompatibilidade de assinatura com defeito era gerada pelo servidor para solicitações do Bolt para clusters com a autenticação do IAM habilitada.
- Correção de um erro do openCypher em que uma consulta usando HTTP keep-alive poderia ser fechada incorretamente se fosse enviada após uma solicitação com falha.
- Correção de um erro do openCypher que poderia causar um erro interno quando uma consulta que gerava um valor constante era enviada.
- Correção de um erro nos detalhes da explicação para que a subconsulta do DFE `Time(ms)` agora some corretamente os tempos de CPU dos operadores na subconsulta do DFE. Pense no seguinte trecho da saída de explicação como exemplo:

```
subQuery1
#######################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments                       #
 Mode # Units In # Units Out # Ratio # Time (ms) #
#######################################################################################
  ...
#######################################################################################
# 1  # 2      # -      # DFEChunkLocalSubQuery # subQuery=...graph#336e.../graph_1 #
 -    # 1        # 1         # 1.00  # 0.38      #
#    #        #        #                       # coordinationTime(ms)=0.026       #
     #          #           #       #           #
#######################################################################################
  ...
subQuery=...graph#336e.../graph_1
#######################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments                       #
 Mode # Units In # Units Out # Ratio # Time (ms) #
#######################################################################################
# 0  # 1      # -      # DFESolutionInjection  # solutions=[?100 -> [-10^^<LONG>]] #
 -    # 0        # 1         # 0.00  # 0.04      #
#    #        #        #                       # outSchema=[?100]                #
     #          #           #       #           #
#######################################################################################
# 1  # 3      # -      # DFERelationalJoin     # joinVars=[]                     #
 -    # 2        # 1         # 0.50  # 0.29      #
#######################################################################################
# 2  # 1      # -      # DFESolutionInjection  # outSchema=[]                   #
 -    # 0        # 1         # 0.00  # 0.01      #
#######################################################################################
```

```
# 3  # -      # -      # DFEDrain              # -                                  #
 -    # 1        # 0         # 0.00  # 0.02      #
#########################################################################################
```

Os tempos de subconsulta na última coluna da tabela inferior somam 0,36 ms (`.04 + .29 + .01 + .02 = .36`). Ao adicionar ao tempo de coordenação dessa subconsulta (`.36 + .026 = .386`), você obtém um resultado próximo ao horário da subconsulta registrada na última coluna da tabela superior, ou seja, `0.38` ms.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.2.0.0, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Versão compatível mais antiga do Gremlin: `3.5.2`
- Versão compatível mais recente do Gremlin: `3.5.4`
- openCypher versão: `Neptune-9.0.20190305-1.0`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.2.0.0 do mecanismo

Como essa é uma versão principal do mecanismo, não há atualização automática para ela.

Somente é possível atualizar para a versão `1.2.0.0` manualmente a partir da versão de patch mais recente da versão 1.1.1.0 do mecanismo. É necessário primeiro atualizar as versões anteriores do mecanismo para a versão mais recente de `1.1.1.0` para que seja possível atualizá-las para `1.2.0.0`.

Portanto, antes de tentar atualizar para esta versão, confirme se você está executando a versão de patch mais recente da versão `1.1.1.0`. Se não estiver, comece atualizando para a versão de patch mais recente da `1.1.1.0`.

Antes da atualização, também é necessário recriar qualquer grupo de parâmetros de cluster de banco de dados personalizado que você estava usando com a versão anterior, usando a família de grupos de parâmetros `neptune1.2`. Consulte Grupos de parâmetros do Amazon Neptune para obter mais informações.

Se você atualizar primeiro para a versão `1.1.1.0` e logo depois para `1.2.0.0`, poderá encontrar um erro como o seguinte:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída (consulte Manter o cluster de banco de dados do Amazon Neptune).

Atualizar para esta versão

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para essa versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Em vez de `--apply-immediately`, é possível especificar `--no-apply-immediately`. Para realizar uma atualização de versão principal, é necessário usar o parâmetro allow-major-version-upgrade. Além disso, não se esqueça de incluir a versão do mecanismo ou ele poderá ser atualizado para outra versão.

Se o cluster usar um grupo de parâmetros de cluster personalizado, não se esqueça de incluir este parâmetro para especificá-lo:

```
--db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Da mesma forma, se alguma instância no cluster usar um grupo de parâmetros de banco de dados personalizado, não se esqueça de incluir este parâmetro para especificá-lo:

```
--db-instance-parameter-group-name (name of the custom instance parameter group)
```

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.2.0.0.R4 (29/09/2023)

Desde 29/09/2023, a versão 1.2.0.0.R4 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

> **Note**
>
> Se estiver fazendo a atualização de uma versão do mecanismo anterior à 1.2.0.0:
>
> - A versão 1.2.0.0 do mecanismo introduziu um novo formato para grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados. Como resultado, se você estiver atualizando de uma versão de mecanismo anterior à 1.2.0.0 para a versão 1.2.0.0 ou posterior, deverá recriar todos os grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados existentes usando a família de grupos de parâmetros `neptune1.2`. As versões anteriores usavam a família de grupos de parâmetros `neptune1`, e esses grupos de parâmetros não funcionarão com a versão 1.2.0.0 e posterior. Consulte Grupos de parâmetros do Amazon Neptune para obter mais informações.

- A versão 1.2.0.0 do mecanismo também introduziu um novo formato para undo logs. Como resultado, todos os undo logs criados por uma versão anterior do mecanismo devem ser eliminados e a métrica `UndoLogsListSize` do CloudWatch deve cair para zero para que seja possível iniciar qualquer atualização de uma versão anterior à 1.2.0.0. Se houver muitos registros de undo logs (duzentos mil ou mais) ao tentar iniciar uma atualização, a tentativa de atualização poderá expirar enquanto aguarda a conclusão da limpeza dos undo logs.

  É possível acelerar a taxa de limpeza atualizando a instância de gravador do cluster, que é onde a limpeza ocorre. Fazer isso antes de tentar realizar a atualização pode reduzir o número de undo logs antes de começar. Aumentar o tamanho do gravador para um tipo de instância 24XL pode aumentar a taxa de limpeza para mais de um milhão de registros por hora.

  Se a métrica `UndoLogsListSize` do CloudWatch for extremamente grande, abrir um caso de suporte pode ajudar a examinar estratégias adicionais para reduzi-la.
- Por fim, houve uma alteração significativa na versão 1.2.0.0. Ela afeta o código anterior que usava o protocolo Bolt com autenticação do IAM. A partir da versão 1.2.0.0, o Bolt precisa de um caminho de recursos para a assinatura do IAM. Em Java, a definição do caminho de recursos pode ser assim: `request.setResourcePath("/openCypher"));`. Em outras linguagens, o `/openCypher` pode ser anexado ao URI do endpoint. Consulte Usar o protocolo Bolt para ver exemplos.

Melhorias nesta versão do mecanismo

- Adição de um parâmetro `enableInterContainerTrafficEncryption` a todas as APIs do Neptune ML, que você pode usar para habilitar e desabilitar a criptografia de tráfego entre contêineres em trabalhos de treinamento ou ajuste de hiperparâmetros.
- Para clusters de banco de dados sem servidor, foram alteradas a configuração de capacidade mínima para 1,0 NCU e a configuração máxima válida mais baixa para 2,5 NCUs. Consulte Escalabilidade da capacidade em um cluster de banco de dados do Neptune Serverless *(antes do lançamento, essa alteração também precisa ser refletida na página sem servidor).*

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro do openCypher em que as consultas de atualização e retorno não tratavam `orderBy`, `limit` e `skip` corretamente.
- Correção de um erro do openCypher que permitia que parâmetros contidos em uma solicitação fossem substituídos por parâmetros contidos em outra solicitação simultânea.
- Correção de um erro do openCypher no tratamento de transações do Bolt.
- Correção dos problemas de exatidão do Gremlin para consultas do DFE com `limit` como percurso filho de etapas que não são de junção retornando ao Tinkerpop. Por exemplo, para consultas como esta:

```
g.withSideEffect('Neptune#useDFE', true)
 .V()
 .as("a")
 .select("a")
 .by(out()
 .limit(1))
```

- Correção de um erro do Gremlin em que uma consulta falhava porque continha muitas etapas do TinkerPop e não era limpa.
- Correção de um erro do Gremlin em que `order()` não classificava corretamente as saídas de string quando algumas delas continham um caractere de espaço.
- Correção de um erro do Gremlin em que um vazamento de transação podia ocorrer quando uma consulta era enviada como uma string e continha `GroupCountStep`.
- Correção de um erro do Gremlin em que ocorria um vazamento de transação ao conferir o endpoint de status da consulta do Gremlin para consultas com predicados em percursos secundários para etapas não processadas de modo nativo.
- Correção de um erro do Gremlin em que adicionar uma borda e suas propriedades seguidas por `inV()` ou `outV()` causava uma `InternalFailureException`.
- Correção de um erro de simultaneidade na camada de armazenamento.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.2.0.0.R4, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Versão compatível mais antiga do Gremlin: `3.5.2`

- Versão compatível mais recente do Gremlin: `3.5.6`
- openCypher versão: `Neptune-9.0.20190305-1.0`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.2.0.0.R4 do mecanismo

O cluster será atualizado com essa versão de patch automaticamente durante a próxima janela de manutenção se você estiver executando a versão `1.2.0.0` do mecanismo.

Somente é possível atualizar para a versão `1.2.0.0` manualmente a partir da versão de patch mais recente da versão 1.1.1.0 do mecanismo. É necessário primeiro atualizar as versões anteriores do mecanismo para a versão mais recente de `1.1.1.0` para que seja possível atualizá-las para `1.2.0.0`.

Se você atualizar primeiro para a versão `1.1.1.0` e logo depois para `1.2.0.0`, poderá encontrar um erro como o seguinte:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Atualizar para esta versão

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para essa versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Em vez de `--apply-immediately`, é possível especificar `--no-apply-immediately`. Para realizar uma atualização de versão principal, é necessário usar o parâmetro allow-major-version-upgrade. Além disso, não se esqueça de incluir a versão do mecanismo ou ele poderá ser atualizado para outra versão.

Se o cluster usar um grupo de parâmetros de cluster personalizado, não se esqueça de incluir este parâmetro para especificá-lo:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Da mesma forma, se alguma instância no cluster usar um grupo de parâmetros de banco de dados personalizado, não se esqueça de incluir este parâmetro para especificá-lo:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.2.0.0.R3 (15/12/2022)

Desde 15/12/2022, a versão 1.2.0.0.R3 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

> **Note**
>
> Se estiver fazendo a atualização de uma versão do mecanismo anterior à 1.2.0.0:

- A versão 1.2.0.0 do mecanismo introduziu um novo formato para grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados. Como resultado, se você estiver atualizando de uma versão de mecanismo anterior à 1.2.0.0 para a versão 1.2.0.0 ou posterior, deverá recriar todos os grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados existentes usando a família de grupos de parâmetros `neptune1.2`. As versões anteriores usavam a família de grupos de parâmetros `neptune1`, e esses grupos de parâmetros não funcionarão com a versão 1.2.0.0 e posterior. Consulte Grupos de parâmetros do Amazon Neptune para obter mais informações.
- A versão 1.2.0.0 do mecanismo também introduziu um novo formato para undo logs. Como resultado, todos os undo logs criados por uma versão anterior do mecanismo devem ser eliminados e a métrica `UndoLogsListSize` do CloudWatch deve cair para zero para que seja possível iniciar qualquer atualização de uma versão anterior à 1.2.0.0. Se houver muitos registros de undo logs (duzentos mil ou mais) ao tentar iniciar uma atualização, a tentativa de atualização poderá expirar enquanto aguarda a conclusão da limpeza dos undo logs.

  É possível acelerar a taxa de limpeza atualizando a instância de gravador do cluster, que é onde a limpeza ocorre. Fazer isso antes de tentar realizar a atualização pode reduzir o número de undo logs antes de começar. Aumentar o tamanho do gravador para um tipo de instância 24XL pode aumentar a taxa de limpeza para mais de um milhão de registros por hora.

  Se a métrica `UndoLogsListSize` do CloudWatch for extremamente grande, abrir um caso de suporte pode ajudar a examinar estratégias adicionais para reduzi-la.
- Por fim, houve uma alteração significativa na versão 1.2.0.0. Ela afeta o código anterior que usava o protocolo Bolt com autenticação do IAM. A partir da versão 1.2.0.0, o Bolt precisa de um caminho de recursos para a assinatura do IAM. Em Java, a definição do caminho de recursos pode ser assim: `request.setResourcePath("/openCypher"));`. Em outras linguagens, o `/openCypher` pode ser anexado ao URI do endpoint. Consulte Usar o protocolo Bolt para ver exemplos.

Melhorias nesta versão do mecanismo

- Melhoria no desempenho das consultas do openCypher que envolvem `MERGE` e `OPTIONAL MATCH`.
- Melhoria no desempenho das consultas do openCypher que envolvem `UNWIND` em uma lista de mapas de valores literais.
- Melhoria no desempenho das consultas do openCypher que têm um filtro `IN` para `id`. Por exemplo:

```
MATCH (n) WHERE id(n) IN ['1', '2', '3'] RETURN n
```

- Melhorias no desempenho e correções para vários operadores do Gremlin, incluindo `repeat`, `coalesce`, `store` e `aggregate`.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro do openCypher em que as consultas geravam uma string, `"null"`, em vez de um valor nulo em Bolt e SPARQL-JSON.
- Correção de um erro do openCypher para interpretar o tipo de parâmetro corretamente quando o valor é uma lista ou uma lista de mapas.
- Correção de um erro no log de auditoria que fazia com que informações desnecessárias fossem registradas e determinados campos ficassem ausentes nos logs.
- Correção de um erro no log de auditoria em que o ARN do IAM das solicitações HTTP para um cluster de banco de dados habilitado para IAM não era registrado.
- Correção de um erro no cache de pesquisa para limitar a memória incremental usada para gravações no cache.
- Correção de um erro do cache de pesquisa que envolvia a configuração do modo somente leitura para o cache de pesquisa quando as gravações falhavam.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.2.0.0.R3, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Versão compatível mais antiga do Gremlin: `3.5.2`
- Versão compatível mais recente do Gremlin: `3.5.4`

- openCypher versão: `Neptune-9.0.20190305-1.0`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.2.0.0.R3 do mecanismo

O cluster será atualizado com essa versão de patch automaticamente durante a próxima janela de manutenção se você estiver executando a versão `1.2.0.0` do mecanismo.

Somente é possível atualizar para a versão `1.2.0.0` manualmente a partir da versão de patch mais recente da versão 1.1.1.0 do mecanismo. É necessário primeiro atualizar as versões anteriores do mecanismo para a versão mais recente de `1.1.1.0` para que seja possível atualizá-las para `1.2.0.0`.

Se você atualizar primeiro para a versão `1.1.1.0` e logo depois para `1.2.0.0`, poderá encontrar um erro como o seguinte:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Atualizar para esta versão

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para essa versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Em vez de `--apply-immediately`, é possível especificar `--no-apply-immediately`. Para realizar uma atualização de versão principal, é necessário usar o parâmetro allow-major-version-upgrade. Além disso, não se esqueça de incluir a versão do mecanismo ou ele poderá ser atualizado para outra versão.

Se o cluster usar um grupo de parâmetros de cluster personalizado, não se esqueça de incluir este parâmetro para especificá-lo:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Da mesma forma, se alguma instância no cluster usar um grupo de parâmetros de banco de dados personalizado, não se esqueça de incluir este parâmetro para especificá-lo:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.2.0.0.R2 (14/10/2022)

Desde 14/10/2022, a versão 1.2.0.0.R2 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

> **Note**
>
> Se estiver fazendo a atualização de uma versão do mecanismo anterior à 1.2.0.0:

- A versão 1.2.0.0 do mecanismo introduziu um novo formato para grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados. Como resultado, se você estiver atualizando de uma versão de mecanismo anterior à 1.2.0.0 para a versão 1.2.0.0 ou posterior, deverá recriar todos os grupos de parâmetros personalizados e grupos de parâmetros de cluster personalizados existentes usando a família de grupos de parâmetros `neptune1.2`. As versões anteriores usavam a família de grupos de parâmetros `neptune1`, e esses grupos de parâmetros não funcionarão com a versão 1.2.0.0 e posterior. Consulte Grupos de parâmetros do Amazon Neptune para obter mais informações.
- A versão 1.2.0.0 do mecanismo também introduziu um novo formato para undo logs. Como resultado, todos os undo logs criados por uma versão anterior do mecanismo devem ser eliminados e a métrica `UndoLogsListSize` do CloudWatch deve cair para zero para que seja possível iniciar qualquer atualização de uma versão anterior à 1.2.0.0. Se houver muitos registros de undo logs (duzentos mil ou mais) ao tentar iniciar uma atualização, a tentativa de atualização poderá expirar enquanto aguarda a conclusão da limpeza dos undo logs.

  É possível acelerar a taxa de limpeza atualizando a instância de gravador do cluster, que é onde a limpeza ocorre. Fazer isso antes de tentar realizar a atualização pode reduzir o número de undo logs antes de começar. Aumentar o tamanho do gravador para um tipo de instância 24XL pode aumentar a taxa de limpeza para mais de um milhão de registros por hora.

  Se a métrica `UndoLogsListSize` do CloudWatch for extremamente grande, abrir um caso de suporte pode ajudar a examinar estratégias adicionais para reduzi-la.
- Por fim, houve uma alteração significativa na versão 1.2.0.0. Ela afeta o código anterior que usava o protocolo Bolt com autenticação do IAM. A partir da versão 1.2.0.0, o Bolt precisa de um caminho de recursos para a assinatura do IAM. Em Java, a definição do caminho de recursos pode ser assim: `request.setResourcePath("/openCypher"));`. Em outras linguagens, o `/openCypher` pode ser anexado ao URI do endpoint. Consulte Usar o protocolo Bolt para ver exemplos.

Melhorias nesta versão do mecanismo

- Melhoria no desempenho de consultas `order-by` do Gremlin. As consultas do Gremlin com um `order-by` no final de uma `NeptuneGraphQueryStep` agora usam um tamanho de bloco maior para melhor desempenho. Isso não se aplica a `order-by` em um nó interno (não raiz) do plano de consulta.
- Melhoria do desempenho de consultas de atualização do Gremlin. Agora, vértices e bordas devem ser bloqueados contra exclusão ao adicionar bordas ou propriedades. Essa alteração elimina bloqueios duplicados em uma transação, o que melhora o desempenho.
- Melhoria no desempenho das consultas do Gremlin que usam `dedup()` em uma subconsulta `repeat()` enviando `dedup` à camada de execução nativa.
- Adição da dica de consulta `Neptune#cardinalityEstimates` do Gremlin. Quando definido como `false`, isso desabilita as estimativas de cardinalidade.
- Adição de mensagens de erro fáceis de usar para erros de autenticação do IAM. Agora, essas mensagens mostram o ARN do usuário ou do perfil do IAM, o ARN do recurso e uma lista de ações não autorizadas da solicitação. A lista de ações não autorizadas ajuda você a ver o que pode estar faltando ou é explicitamente negado na política do IAM que você está usando.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro de exatidão do Gremlin envolvendo a conversão de `WherePredicateStep`, em que o mecanismo de consulta do Neptune estava produzindo resultados incorretos para consultas que usavam `where(P.neq('x'))` e as variações dela.
- Correção de um erro do Gremlin em que o uso de `PartitionStrategy` após a atualização para o TinkerPop 3.5 gerava incorretamente um erro com a mensagem “PartitionStrategy não funciona com percursos anônimos”, o que impedia a execução do percurso.
- Correção de vários erros do Gremlin relacionados ao `joinTime` de uma junção final e às estatísticas nos subgrupos `Project.ASK`.
- Correção de um erro do openCypher na cláusula `MERGE` que, em alguns casos, causava a criação duplicada de nós e bordas.
- Correção de um erro de transação em que uma sessão podia inserir dados de grafos e confirmar mesmo quando as inserções simultâneas correspondentes do dicionário eram revertidas.
- Correção de um erro do carregador em massa que causava regressões de desempenho sob cargas de inserção pesadas.

- Correção de um erro no tratamento de consultas do SPARQL que contêm `(NOT) EXISTS` em uma cláusula `OPTIONAL` em que, em alguns casos, os resultados da consulta ficavam ausentes.
- Correção de um erro em que os drivers pareciam travar nos casos em que as solicitações eram canceladas devido a um tempo limite antes do início da avaliação. Era possível entrar nesse estado se todos os threads de processamento de consultas no servidor fossem consumidos enquanto os tempos limite ocorressem nos itens na fila de solicitações. Como os tempos limite da fila de solicitações não estavam enviando mensagens imediatamente, as respostas pareciam pendentes para o cliente.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.2.0.0.R2, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Versão compatível mais antiga do Gremlin: `3.5.2`
- Versão compatível mais recente do Gremlin: `3.5.4`
- openCypher versão: `Neptune-9.0.20190305-1.0`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.2.0.0.R2 do mecanismo

O cluster será atualizado com essa versão de patch automaticamente durante a próxima janela de manutenção se você estiver executando a versão `1.2.0.0` do mecanismo.

Somente é possível atualizar para a versão `1.2.0.0` manualmente a partir da versão de patch mais recente da versão 1.1.1.0 do mecanismo. É necessário primeiro atualizar as versões anteriores do mecanismo para a versão mais recente de `1.1.1.0` para que seja possível atualizá-las para `1.2.0.0`.

Se você atualizar primeiro para a versão `1.1.1.0` e logo depois para `1.2.0.0`, poderá encontrar um erro como o seguinte:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Atualizar para esta versão

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para essa versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Em vez de `--apply-immediately`, é possível especificar `--no-apply-immediately`. Para realizar uma atualização de versão principal, é necessário usar o parâmetro allow-major-version-upgrade. Além disso, não se esqueça de incluir a versão do mecanismo ou ele poderá ser atualizado para outra versão.

Se o cluster usar um grupo de parâmetros de cluster personalizado, não se esqueça de incluir este parâmetro para especificá-lo:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Da mesma forma, se alguma instância no cluster usar um grupo de parâmetros de banco de dados personalizado, não se esqueça de incluir este parâmetro para especificá-lo:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.1.1.0 (19/04/2022)

Desde 19/04/2022, a versão 1.1.1.0 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Important

A atualização para esta versão do mecanismo a partir de uma versão anterior à **`1.1.0.0`** também aciona uma atualização do sistema operacional em todas as instâncias do cluster de banco de dados. Como as solicitações de gravação ativas que ocorrem durante a atualização do sistema operacional não serão processadas, você deve pausar todas as workloads de gravação no cluster que está sendo atualizado, incluindo carregamentos de dados em massa, antes de iniciar a atualização.

Para concluir a atualização com êxito, todas as sub-redes em cada zona de disponibilidade (AZ) devem ter pelo menos um endereço IP disponível por instância do Neptune. Por exemplo, se houver uma instância de gravador e duas instâncias de leitor na sub-rede 1 e duas instâncias de leitor na sub-rede 2, a sub-rede 1 deverá ter pelo menos três endereços IP livres e a sub-rede 2 deverá ter pelo menos dois endereços IP livres antes de iniciar a atualização.

No início da atualização, o Neptune gera um snapshot com um nome composto de `preupgrade` seguido por um identificador gerado automaticamente com base nas informações do cluster de banco de dados. Você não será cobrado por esse snapshot e poderá usá-lo para restaurar o cluster de banco de dados se algo der errado durante o processo de atualização.

Quando a atualização do mecanismo em si for concluída, a nova versão do mecanismo estará disponível brevemente no sistema operacional antigo, mas em menos de cinco minutos todas as instâncias do cluster iniciarão simultaneamente uma atualização do sistema

operacional. O cluster de banco de dados ficará indisponível nesse momento por alguns minutos. Você poderá retomar as workloads de gravação após a conclusão da atualização. Esse processo gera os seguintes eventos:

- Mensagens de eventos por cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- Mensagens de eventos por instância:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Versões de patch subsequentes para esta versão

- Versão de manutenção: 1.1.1.0.R2 (16/05/2022)
- Versão: 1.1.1.0.R3 (07/06/2022)
- Versão: 1.1.1.0.R4 (23/06/2022)
- Versão: 1.1.1.0.R5 (21/07/2022)
- Versão: 1.1.1.0.R6 (23/09/2022)
- Versão: 1.1.1.0.R7 (23/01/2023)

Novos recursos nesta versão do mecanismo

- A linguagem de consulta openCypher agora está disponível para o público para uso em produção.

> **Warning**
>
> Há uma alteração significativa nesta versão para o código que usa o openCypher com autenticação do IAM. Na pré-visualização do Neptune para openCypher, a string do host na assinatura do IAM incluía o protocolo, como `bolt://`, desta forma:

```
"Host":"bolt://(host URL):(port)"
```

A partir desta versão do mecanismo, o protocolo deve ser omitido:

```
"Host":"(host URL):(port)"
```

Consulte Usar o protocolo Bolt para ver exemplos.

- Adição de suporte para o TinkerPop `3.5.2`. Entre as alterações nesta versão estão o suporte para transações remotas e o suporte a bytecode para sessões (usando `g.tx`) e a adição da função `datetime()` à linguagem Gremlin.

> **Warning**
>
> Existem algumas alterações importantes introduzidas no TinkerPop 3.5.0, 3.5.1 e 3.5.2 que podem afetar o código do Gremlin. Por exemplo, usar percursos gerados por um GraphTraversalSource como filhos dessa forma não funcionará mais: `g.V().union(identity(), g.V())`.
> Agora, em vez disso, use um percurso anônimo como este: `g.V().union(identity(), __.V())`.

- Adição de suporte para chaves de condição globais da AWS que você pode usar nas políticas de acesso a dados do IAM que controlam o acesso aos dados armazenados no Neptune, um cluster de banco de dados do Neptune.
- O mecanismo de consulta DFE do Neptune agora está disponível ao público para uso em produção com a linguagem de consulta openCypher, mas ainda não para consultas do Gremlin e do SPARQL. Agora você o habilita usando o próprio parâmetro de instância neptune_dfe_query_engine em vez do parâmetro de modo de laboratório.

Melhorias nesta versão do mecanismo

- Adição de novos atributos ao openCypher, como suporte a consultas parametrizadas, armazenamento em cache de árvore de sintaxe abstrata (AST) para consultas parametrizadas, melhorias no caminho de comprimento variável (VLP) e novos operadores e cláusulas. Consulte Conformidade com a especificação OpenCypher no Amazon Neptune para conhecer o nível atual de suporte a linguagens.

- Melhorias significativas no desempenho do openCypher para workloads simples de leitura e gravação, ocasionando maior throughput em comparação com a versão 1.1.0.0.
- Remoção das limitações bidirecionais e de profundidade do openCypher que lidam com caminhos de comprimento variável.
- Suporte completo no mecanismo DFE para os predicados `within` e `without` do Gremlin, incluindo casos em que eles são combinados com outros operadores de predicados. Por exemplo:

```
g.V().has('age', within(12, 15, 18).or(gt(30)))
```

- Suporte estendido no mecanismo DFE para a etapa `order` do Gremlin quando o escopo é global (ou seja, não `order(local)`) e quando os moduladores `by()` não são usados. Por exemplo, esta consulta agora teria suporte para DFE:

```
 g.V().values("age").order()
```

- Adição de um campo `isLastOp` ao formato de resposta do log de alterações de fluxos do Neptune para indicar que um registro é a última operação na transação.
- Melhoria significativa do desempenho do registro em log de auditoria e redução da latência quando o registro em log de auditoria está habilitado.
- Conversão de consultas de bytecode e HTTP WebSocket do Gremlin em um formato legível pelo usuário nos logs de auditoria. Agora, as consultas podem ser copiadas diretamente dos logs de auditoria para serem executadas nos blocos de anotações Neptune e em outros lugares. Observe que essa alteração no formato atual do log de auditoria constitui uma alteração significativa.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro raro do Gremlin em que nenhum resultado era exibido ao usar as etapas `filter()` e `count()` aninhadas em combinação, como nesta consulta:

```
g.V("1").filter(out("knows")
        .filter(in("knows")
        .hasId("notExists"))
        .count())
```

- Correção de um erro do Gremlin em que um erro era gerado ao usar um vértice armazenado por uma etapa agregada em um percurso `to()` ou `from()` com uma etapa `addE`. Veja um exemplo de consulta desse:

```
g.V("id").aggregate("v").out().addE().to(select("v").unfold()))
```

- Correção de um erro do Gremlin em que a etapa `not` falhava em casos de borda ao usar o mecanismo DFE. Por exemplo:

```
g.V().not(V())
```

- Correção de erro do Gremlin em que valores `sideEffect` não estavam disponíveis em percursos `to()` e `from()`.
- Correção de um erro que ocasionalmente fazia com que uma redefinição rápida acionasse um failover de instância.
- Correção de um erro do carregador em massa em que uma transação com falha não era fechada antes do início do próximo trabalho de carregamento.
- Correção de um erro do carregador em massa em que uma condição de pouca memória poderia causar travamento do sistema.
- Adição de uma nova tentativa para corrigir um erro do carregador em massa em que o carregador não esperava o suficiente para que as credenciais do IAM ficassem disponíveis após um failover.
- Correção de um erro em que o cache interno de credenciais não estava sendo limpo adequadamente para endpoints que não eram de consulta, como o endpoint de `status`.
- Correção de um erro de fluxos para garantir que os números de sequência de confirmação de fluxos sejam ordenados corretamente.
- Correção de um erro em que conexões de longa execução eram encerradas antes de dez dias em clusters habilitados para IAM.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.1.1.0, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Versão compatível mais antiga do Gremlin: `3.5.2`
- Versão compatível mais recente do Gremlin: `3.5.4`
- openCypher versão: `Neptune-9.0.20190305-1.0`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.1.1.0 do mecanismo

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão. Observe que as versões anteriores ao mecanismo de versão principal (1.1.0.0) levarão mais tempo para serem atualizadas para esta versão.

A atualização para esta versão não será automática.

Atualizar para esta versão

Important

A atualização para esta versão do mecanismo a partir de qualquer versão anterior à **`1.1.0.0`** também aciona uma atualização do sistema operacional em todas as instâncias no cluster de banco de dados. Como as solicitações de gravação ativas que ocorrem durante a atualização do sistema operacional não serão processadas, você deve pausar todas as workloads de gravação no cluster que está sendo atualizado, incluindo carregamentos de dados em massa, antes de iniciar a atualização.
No início da atualização, o Neptune gera um snapshot com um nome composto de `preupgrade` seguido por um identificador gerado automaticamente com base nas informações do cluster de banco de dados. Você não será cobrado por esse snapshot e poderá usá-lo para restaurar o cluster de banco de dados se algo der errado durante o processo de atualização.
Quando a atualização do mecanismo em si for concluída, a nova versão do mecanismo estará disponível brevemente no sistema operacional antigo, mas em menos de cinco minutos todas as instâncias do cluster iniciarão simultaneamente uma atualização do sistema operacional. O cluster de banco de dados ficará indisponível nesse momento por cerca de seis minutos. Você poderá retomar as workloads de gravação após a conclusão da atualização.
Esse processo gera os seguintes eventos:

- Mensagens de eventos por cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Mensagens de eventos por instância:
  - `Applying off-line patches to DB instance`

- DB instance shutdown
- Finished applying off-line patches to DB instance
- DB instance restarted

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para essa versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine neptune \
    --engine-version 1.1.1.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine neptune ^
    --engine-version 1.1.1.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Em vez de `--apply-immediately`, é possível especificar `--no-apply-immediately`. Para realizar uma atualização de versão principal, é necessário usar o parâmetro allow-major-version-upgrade. Além disso, não se esqueça de incluir a versão do mecanismo ou ele poderá ser atualizado para outra versão.

Se o cluster usar um grupo de parâmetros de cluster personalizado, não se esqueça de incluir este parâmetro para especificá-lo:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Da mesma forma, se alguma instância no cluster usar um grupo de parâmetros de banco de dados personalizado, não se esqueça de incluir este parâmetro para especificá-lo:

```
--db-instance-parameter-group-name (name of the custom instance parameter group)
```

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
> ```

```
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.1.1.0.R7 (23/01/2023)

Desde 23/01/2023, a versão 1.1.1.0.R7 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Important

A atualização para esta versão do mecanismo a partir de uma versão anterior à **`1.1.0.0`** também aciona uma atualização do sistema operacional em todas as instâncias do cluster de banco de dados. Como as solicitações de gravação ativas que ocorrem durante a atualização do sistema operacional não serão processadas, você deve pausar todas as workloads de gravação no cluster que está sendo atualizado, incluindo carregamentos de dados em massa, antes de iniciar a atualização.

Para concluir a atualização com êxito, todas as sub-redes em cada zona de disponibilidade (AZ) devem ter pelo menos um endereço IP disponível por instância do Neptune. Por exemplo, se houver uma instância de gravador e duas instâncias de leitor na sub-rede 1 e duas instâncias de leitor na sub-rede 2, a sub-rede 1 deverá ter pelo menos três endereços IP livres e a sub-rede 2 deverá ter pelo menos dois endereços IP livres antes de iniciar a atualização.

No início da atualização, o Neptune gera um snapshot com um nome composto de `preupgrade` seguido por um identificador gerado automaticamente com base nas informações do cluster de banco de dados. Você não será cobrado por esse snapshot e poderá usá-lo para restaurar o cluster de banco de dados se algo der errado durante o processo de atualização.

Quando a atualização do mecanismo em si for concluída, a nova versão do mecanismo estará disponível brevemente no sistema operacional antigo, mas em menos de cinco minutos todas as instâncias do cluster iniciarão simultaneamente uma atualização do sistema operacional. O cluster de banco de dados ficará indisponível nesse momento por alguns minutos. Você poderá retomar as workloads de gravação após a conclusão da atualização. Esse processo gera os seguintes eventos:

- Mensagens de eventos por cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- Mensagens de eventos por instância:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Melhorias nesta versão do mecanismo

- Melhoria no desempenho das consultas do openCypher que envolvem `MERGE` e `OPTIONAL MATCH`.
- Melhoria no desempenho das consultas do openCypher que envolvem `UNWIND` de uma lista de mapas de valores literais.
- Melhoria no desempenho das consultas do openCypher que têm um filtro `IN` para `id`. Por exemplo:

```
MATCH (n) WHERE id(n) IN ['1', '2', '3'] RETURN n
```

- Melhorias no desempenho e correções para vários operadores do Gremlin, incluindo `repeat`, `coalesce`, `store` e `aggregate`.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro do openCypher em que uma solicitação usando HTTP keep-alive poderia ser fechada incorretamente se fosse enviada após uma solicitação com falha.
- Correção de um erro do openCypher em que o tipo de parâmetro nem sempre era interpretado corretamente para uma lista ou uma lista de mapas.
- Correção de um erro do openCypher em que as consultas geravam uma string, `"null"`, em vez de um valor nulo em Bolt e SPARQL-JSON.
- Correção de códigos e mensagens de erro do openCypher para falhas de tempo limite de consulta e erros de memória insuficiente.
- Correção de um erro do Gremlin que fazia com que `valueMap()` não fosse otimizado em um percurso `by()` no mecanismo do DFE.
- Correção de um problema com a lógica do detector de deadlock que ocasionalmente fazia com que o mecanismo parasse de responder.
- Correção de um erro no log de auditoria que fazia com que informações desnecessárias fossem registradas e determinados campos ficassem ausentes nos logs.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.1.1.0.R7, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Versão compatível mais antiga do Gremlin: `3.5.2`
- Versão compatível mais recente do Gremlin: `3.5.3`
- openCypher versão: `Neptune-9.0.20190305-1.0`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.1.1.0.R7 do mecanismo

O cluster será atualizado com essa versão de patch automaticamente durante a próxima janela de manutenção se você estiver executando a versão `1.1.1.0` do mecanismo.

Atualizar para esta versão

> **Important**
>
> A atualização para esta versão do mecanismo a partir de qualquer versão anterior à **`1.1.0.0`** também aciona uma atualização do sistema operacional em todas as instâncias no cluster de banco de dados. Como as solicitações de gravação ativas que ocorrem durante a atualização do sistema operacional não serão processadas, você deve pausar todas as workloads de gravação no cluster que está sendo atualizado, incluindo carregamentos de dados em massa, antes de iniciar a atualização.
>
> No início da atualização, o Neptune gera um snapshot com um nome composto de `preupgrade` seguido por um identificador gerado automaticamente com base nas informações do cluster de banco de dados. Você não será cobrado por esse snapshot e poderá usá-lo para restaurar o cluster de banco de dados se algo der errado durante o processo de atualização.
>
> Quando a atualização do mecanismo em si for concluída, a nova versão do mecanismo estará disponível brevemente no sistema operacional antigo, mas em menos de cinco minutos todas as instâncias do cluster iniciarão simultaneamente uma atualização do sistema operacional. O cluster de banco de dados ficará indisponível nesse momento por cerca de seis minutos. Você poderá retomar as workloads de gravação após a conclusão da atualização.
>
> Esse processo gera os seguintes eventos:
>
> - Mensagens de eventos por cluster:
>   - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
>   - `Database cluster major version has been upgraded`
> - Mensagens de eventos por instância:
>   - `Applying off-line patches to DB instance`
>   - `DB instance shutdown`
>   - `Finished applying off-line patches to DB instance`
>   - `DB instance restarted`

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para essa versão, ele estará elegível para ser atualizado agora.

Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.1.1.0.R6 (23/09/2022)

Desde 23/09/2022, a versão 1.1.1.0.R6 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

> **Important**
>
> A atualização para esta versão do mecanismo a partir de uma versão anterior à **`1.1.0.0`** também aciona uma atualização do sistema operacional em todas as instâncias do cluster de banco de dados. Como as solicitações de gravação ativas que ocorrem durante a

atualização do sistema operacional não serão processadas, você deve pausar todas as workloads de gravação no cluster que está sendo atualizado, incluindo carregamentos de dados em massa, antes de iniciar a atualização.
Para concluir a atualização com êxito, todas as sub-redes em cada zona de disponibilidade (AZ) devem ter pelo menos um endereço IP disponível por instância do Neptune. Por exemplo, se houver uma instância de gravador e duas instâncias de leitor na sub-rede 1 e duas instâncias de leitor na sub-rede 2, a sub-rede 1 deverá ter pelo menos três endereços IP livres e a sub-rede 2 deverá ter pelo menos dois endereços IP livres antes de iniciar a atualização.
No início da atualização, o Neptune gera um snapshot com um nome composto de `preupgrade` seguido por um identificador gerado automaticamente com base nas informações do cluster de banco de dados. Você não será cobrado por esse snapshot e poderá usá-lo para restaurar o cluster de banco de dados se algo der errado durante o processo de atualização.
Quando a atualização do mecanismo em si for concluída, a nova versão do mecanismo estará disponível brevemente no sistema operacional antigo, mas em menos de cinco minutos todas as instâncias do cluster iniciarão simultaneamente uma atualização do sistema operacional. O cluster de banco de dados ficará indisponível nesse momento por alguns minutos. Você poderá retomar as workloads de gravação após a conclusão da atualização.
Esse processo gera os seguintes eventos:

- Mensagens de eventos por cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Mensagens de eventos por instância:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

> **Note**
>
> Há uma alteração significativa nesta versão para o código que usa o openCypher com autenticação do IAM. Até o momento, a string do host na assinatura do IAM incluía o protocolo, como `bolt://`, por exemplo:
>
> ```
> "Host":"bolt://(host URL):(port)"
> ```
>
> A partir da versão do mecanismo `1.1.1.0`, o protocolo deve ser omitido:
>
> ```
> "Host":"(host URL):(port)"
> ```
>
> Consulte Usar o protocolo Bolt para ver exemplos.

Melhorias nesta versão do mecanismo

- Melhoria no desempenho de consultas `order-by` do Gremlin. As consultas do Gremlin com um `order-by` no final de uma `NeptuneGraphQueryStep` agora usam um tamanho de bloco maior para melhor desempenho. Isso não se aplica a `order-by` em um nó interno (não raiz) do plano de consulta.
- Melhoria do desempenho de consultas de atualização do Gremlin. Agora, vértices e bordas devem ser bloqueados contra exclusão ao adicionar bordas ou propriedades. Essa alteração elimina bloqueios duplicados em uma transação, o que melhora o desempenho.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro do openCypher na cláusula `MERGE` que, em alguns casos, causava a criação duplicada de nós e bordas.
- Correção de um erro no tratamento de consultas do SPARQL que contêm `(NOT) EXISTS` em uma cláusula `OPTIONAL` em que, em alguns casos, os resultados da consulta ficavam ausentes.
- Correção de um erro que atrasava a reinicialização do servidor quando um carregamento em massa estava em andamento.
- Correção de um erro em que um percurso bidirecional do padrão de comprimento variável do openCypher com um filtro na propriedade de relacionamento causava um erro. Um exemplo desse padrão de comprimento variável é `(n)-[r*1..2]->(m)`.

- Correção de um erro relacionado à forma como os dados em cache são enviados de volta ao cliente, que em alguns casos resultava em uma latência inesperadamente longa.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.1.1.0.R6, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Versão compatível mais antiga do Gremlin: `3.5.2`
- Versão compatível mais recente do Gremlin: `3.5.4`
- openCypher versão: `Neptune-9.0.20190305-1.0`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.1.1.0.R6 do mecanismo

O cluster será atualizado com essa versão de patch automaticamente durante a próxima janela de manutenção se você estiver executando a versão `1.1.1.0` do mecanismo.

Atualizar para esta versão

Important

A atualização para esta versão do mecanismo a partir de qualquer versão anterior à **`1.1.0.0`** também aciona uma atualização do sistema operacional em todas as instâncias no cluster de banco de dados. Como as solicitações de gravação ativas que ocorrem durante a atualização do sistema operacional não serão processadas, você deve pausar todas as workloads de gravação no cluster que está sendo atualizado, incluindo carregamentos de dados em massa, antes de iniciar a atualização.

No início da atualização, o Neptune gera um snapshot com um nome composto de `preupgrade` seguido por um identificador gerado automaticamente com base nas informações do cluster de banco de dados. Você não será cobrado por esse snapshot e poderá usá-lo para restaurar o cluster de banco de dados se algo der errado durante o processo de atualização.

Quando a atualização do mecanismo em si for concluída, a nova versão do mecanismo estará disponível brevemente no sistema operacional antigo, mas em menos de cinco minutos todas as instâncias do cluster iniciarão simultaneamente uma atualização do sistema

operacional. O cluster de banco de dados ficará indisponível nesse momento por cerca de seis minutos. Você poderá retomar as workloads de gravação após a conclusão da atualização.
Esse processo gera os seguintes eventos:

- Mensagens de eventos por cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- Mensagens de eventos por instância:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para essa versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias,

ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
```

```
proceeding with the upgrade.
```

Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.1.1.0.R5 (21/07/2022)

Desde 21/07/2022, a versão 1.1.1.0.R5 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Important

A atualização para esta versão do mecanismo a partir de uma versão anterior à **`1.1.0.0`** também aciona uma atualização do sistema operacional em todas as instâncias do cluster de banco de dados. Como as solicitações de gravação ativas que ocorrem durante a atualização do sistema operacional não serão processadas, você deve pausar todas as workloads de gravação no cluster que está sendo atualizado, incluindo carregamentos de dados em massa, antes de iniciar a atualização.

Para concluir a atualização com êxito, todas as sub-redes em cada zona de disponibilidade (AZ) devem ter pelo menos um endereço IP disponível por instância do Neptune. Por exemplo, se houver uma instância de gravador e duas instâncias de leitor na sub-rede 1 e duas instâncias de leitor na sub-rede 2, a sub-rede 1 deverá ter pelo menos três endereços IP livres e a sub-rede 2 deverá ter pelo menos dois endereços IP livres antes de iniciar a atualização.

No início da atualização, o Neptune gera um snapshot com um nome composto de `preupgrade` seguido por um identificador gerado automaticamente com base nas informações do cluster de banco de dados. Você não será cobrado por esse snapshot e poderá usá-lo para restaurar o cluster de banco de dados se algo der errado durante o processo de atualização.

Quando a atualização do mecanismo em si for concluída, a nova versão do mecanismo estará disponível brevemente no sistema operacional antigo, mas em menos de cinco minutos todas as instâncias do cluster iniciarão simultaneamente uma atualização do sistema

operacional. O cluster de banco de dados ficará indisponível nesse momento por alguns minutos. Você poderá retomar as workloads de gravação após a conclusão da atualização. Esse processo gera os seguintes eventos:

- Mensagens de eventos por cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- Mensagens de eventos por instância:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Note

Há uma alteração significativa nesta versão para o código que usa o openCypher com autenticação do IAM. Até o momento, a string do host na assinatura do IAM incluía o protocolo, como `bolt://`, por exemplo:

```
"Host":"bolt://(host URL):(port)"
```

A partir da versão do mecanismo `1.1.1.0`, o protocolo deve ser omitido:

```
"Host":"(host URL):(port)"
```

Consulte Usar o protocolo Bolt para ver exemplos.

Melhorias nesta versão do mecanismo

- Foram realizadas melhorias para oferecer compatibilidade com a detecção de deadlocks.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro que impedia um desligamento limpo dos clusters de banco de dados em determinadas condições.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.1.1.0.R5, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Versão compatível mais antiga do Gremlin: `3.5.2`
- Versão compatível mais recente do Gremlin: `3.5.4`
- openCypher versão: `Neptune-9.0.20190305-1.0`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.1.1.0.R5 do mecanismo

O cluster será atualizado com essa versão de patch automaticamente durante a próxima janela de manutenção se você estiver executando a versão `1.1.1.0` do mecanismo.

Atualizar para esta versão

Important

A atualização para esta versão do mecanismo a partir de qualquer versão anterior à **`1.1.0.0`** também aciona uma atualização do sistema operacional em todas as instâncias no cluster de banco de dados. Como as solicitações de gravação ativas que ocorrem durante a atualização do sistema operacional não serão processadas, você deve pausar todas as workloads de gravação no cluster que está sendo atualizado, incluindo carregamentos de dados em massa, antes de iniciar a atualização.

No início da atualização, o Neptune gera um snapshot com um nome composto de `preupgrade` seguido por um identificador gerado automaticamente com base nas informações do cluster de banco de dados. Você não será cobrado por esse snapshot e poderá usá-lo para restaurar o cluster de banco de dados se algo der errado durante o processo de atualização.

Quando a atualização do mecanismo em si for concluída, a nova versão do mecanismo estará disponível brevemente no sistema operacional antigo, mas em menos de cinco

minutos todas as instâncias do cluster iniciarão simultaneamente uma atualização do sistema operacional. O cluster de banco de dados ficará indisponível nesse momento por cerca de seis minutos. Você poderá retomar as workloads de gravação após a conclusão da atualização.
Esse processo gera os seguintes eventos:

- Mensagens de eventos por cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Mensagens de eventos por instância:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para essa versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
> ```

```
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.1.1.0.R4 (23/06/2022)

Desde 23/06/2022, a versão 1.1.1.0.R4 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Important

A atualização para esta versão do mecanismo a partir de uma versão anterior à **`1.1.0.0`** também aciona uma atualização do sistema operacional em todas as instâncias do cluster de banco de dados. Como as solicitações de gravação ativas que ocorrem durante a atualização do sistema operacional não serão processadas, você deve pausar todas as workloads de gravação no cluster que está sendo atualizado, incluindo carregamentos de dados em massa, antes de iniciar a atualização.

Para concluir a atualização com êxito, todas as sub-redes em cada zona de disponibilidade (AZ) devem ter pelo menos um endereço IP disponível por instância do Neptune. Por exemplo, se houver uma instância de gravador e duas instâncias de leitor na sub-rede 1 e duas instâncias de leitor na sub-rede 2, a sub-rede 1 deverá ter pelo menos três endereços IP livres e a sub-rede 2 deverá ter pelo menos dois endereços IP livres antes de iniciar a atualização.

No início da atualização, o Neptune gera um snapshot com um nome composto de `preupgrade` seguido por um identificador gerado automaticamente com base nas informações do cluster de banco de dados. Você não será cobrado por esse snapshot e poderá usá-lo para restaurar o cluster de banco de dados se algo der errado durante o processo de atualização.

Quando a atualização do mecanismo em si for concluída, a nova versão do mecanismo estará disponível brevemente no sistema operacional antigo, mas em menos de cinco minutos todas as instâncias do cluster iniciarão simultaneamente uma atualização do sistema operacional. O cluster de banco de dados ficará indisponível nesse momento por alguns minutos. Você poderá retomar as workloads de gravação após a conclusão da atualização. Esse processo gera os seguintes eventos:

- Mensagens de eventos por cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Mensagens de eventos por instância:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Note

Há uma alteração significativa nesta versão para o código que usa o openCypher com autenticação do IAM. Até o momento, a string do host na assinatura do IAM incluía o protocolo, como `bolt://`, por exemplo:

```
"Host":"bolt://(host URL):(port)"
```

A partir da versão do mecanismo `1.1.1.0`, o protocolo deve ser omitido:

```
"Host":"(host URL):(port)"
```

Consulte Usar o protocolo Bolt para ver exemplos.

Melhorias nesta versão do mecanismo

- Atualização da configuração dos tipos de instância `x2g`.

- Melhoria no desempenho das quedas de vértices.

Defeitos corrigidos nesta versão do mecanismo

- Correção do erro do Gremlin em que as soluções não mantinham uma ordem estável para uma consulta chamada várias vezes ou em vários leitores para determinados tipos de junções ASK.
- Além disso, reduzimos o escopo de uma alteração na versão anterior que estava causando regressões de desempenho para certos tipos de junções ASK no Gremlin.
- Correção de um erro do Gremlin na etapa `union()` que ocorria quando havia uma entrada de borda e um percurso para um vértice nos percursos secundários.
- Correção de um erro no perfil do Gremlin em que algumas etapas eram relatadas como não otimizadas quando na realidade estavam.
- Correção de um erro do SPARQL em que variáveis usadas em expressões `FILTER` aninhadas em cláusulas `UNION` recebiam informações de escopo inválidas.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.1.1.0.R4, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Versão compatível mais antiga do Gremlin: `3.5.2`
- Versão compatível mais recente do Gremlin: `3.5.4`
- openCypher versão: `Neptune-9.0.20190305-1.0`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.1.1.0.R4 do mecanismo

O cluster será atualizado com essa versão de patch automaticamente durante a próxima janela de manutenção se você estiver executando a versão `1.1.1.0` do mecanismo.

Atualizar para esta versão

> **Important**
>
> A atualização para esta versão do mecanismo a partir de qualquer versão anterior à **`1.1.0.0`** também aciona uma atualização do sistema operacional em todas as instâncias no

cluster de banco de dados. Como as solicitações de gravação ativas que ocorrem durante a atualização do sistema operacional não serão processadas, você deve pausar todas as workloads de gravação no cluster que está sendo atualizado, incluindo carregamentos de dados em massa, antes de iniciar a atualização.
No início da atualização, o Neptune gera um snapshot com um nome composto de `preupgrade` seguido por um identificador gerado automaticamente com base nas informações do cluster de banco de dados. Você não será cobrado por esse snapshot e poderá usá-lo para restaurar o cluster de banco de dados se algo der errado durante o processo de atualização.
Quando a atualização do mecanismo em si for concluída, a nova versão do mecanismo estará disponível brevemente no sistema operacional antigo, mas em menos de cinco minutos todas as instâncias do cluster iniciarão simultaneamente uma atualização do sistema operacional. O cluster de banco de dados ficará indisponível nesse momento por cerca de seis minutos. Você poderá retomar as workloads de gravação após a conclusão da atualização.
Esse processo gera os seguintes eventos:

- Mensagens de eventos por cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Mensagens de eventos por instância:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para essa versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
```

```
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome

que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.1.1.0.R3 (07/06/2022)

Desde 07/06/2022, a versão 1.1.1.0.R3 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

> **Important**
>
> A atualização para esta versão do mecanismo a partir de uma versão anterior à **`1.1.0.0`** também aciona uma atualização do sistema operacional em todas as instâncias do cluster de banco de dados. Como as solicitações de gravação ativas que ocorrem durante a atualização do sistema operacional não serão processadas, você deve pausar todas as workloads de gravação no cluster que está sendo atualizado, incluindo carregamentos de dados em massa, antes de iniciar a atualização.
> No início da atualização, o Neptune gera um snapshot com um nome composto de `preupgrade` seguido por um identificador gerado automaticamente com base nas

informações do cluster de banco de dados. Você não será cobrado por esse snapshot e poderá usá-lo para restaurar o cluster de banco de dados se algo der errado durante o processo de atualização.
Quando a atualização do mecanismo em si for concluída, a nova versão do mecanismo estará disponível brevemente no sistema operacional antigo, mas em menos de cinco minutos todas as instâncias do cluster iniciarão simultaneamente uma atualização do sistema operacional. O cluster de banco de dados ficará indisponível nesse momento por alguns minutos. Você poderá retomar as workloads de gravação após a conclusão da atualização. Esse processo gera os seguintes eventos:

- Mensagens de eventos por cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Mensagens de eventos por instância:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Note

Há uma alteração significativa nesta versão para o código que usa o openCypher com autenticação do IAM. Até o momento, a string do host na assinatura do IAM incluía o protocolo, como `bolt://`, por exemplo:

```
"Host":"bolt://(host URL):(port)"
```

A partir da versão do mecanismo `1.1.1.0`, o protocolo deve ser omitido:

```
"Host":"(host URL):(port)"
```

Consulte Usar o protocolo Bolt para ver exemplos.

Melhorias nesta versão do mecanismo

- Adição de suporte para os tipos de instância `x2g` equipados com Graviton2, otimizados para workloads com uso intenso de memória. Inicialmente, eles estão disponíveis apenas em quatro Regiões da AWS:
  - Leste dos EUA (Norte da Virgínia) (`us-east-1`)
  - Leste dos EUA (Ohio) (`us-east-2`)
  - Oeste dos EUA (Oregon) (`us-west-2`)
  - Europa (Irlanda) (`eu-west-1`)

  Para obter mais informações, consulte a página Preços do Neptune.
- Melhoria no desempenho das etapas do Gremlin em que vários percursos de bordas ou vértices, pesquisas de propriedades ou pesquisas de rótulos estão envolvidos.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro do Gremlin no processamento da etapa `otherV()` em um percurso secundário.
- Correção de um erro do Gremlin em consultas com `union` que têm apenas etapas de filtro como filhos. Por exemplo:

```
g.V().union(has("name"), out("knows")).out()
```

- Correção de um erro do SPARQL em que variáveis usadas em expressões `FILTER` aninhadas em cláusulas `UNION` recebiam informações de escopo inválidas.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.1.1.0.R3, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Versão compatível mais antiga do Gremlin: `3.5.2`
- Versão compatível mais recente do Gremlin: `3.5.4`
- openCypher versão: `Neptune-9.0.20190305-1.0`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.1.1.0.R3 do mecanismo

O cluster será atualizado com essa versão de patch automaticamente durante a próxima janela de manutenção se você estiver executando a versão `1.1.1.0` do mecanismo.

Atualizar para esta versão

Important

A atualização para esta versão do mecanismo a partir de qualquer versão anterior à **`1.1.0.0`** também aciona uma atualização do sistema operacional em todas as instâncias no cluster de banco de dados. Como as solicitações de gravação ativas que ocorrem durante a atualização do sistema operacional não serão processadas, você deve pausar todas as workloads de gravação no cluster que está sendo atualizado, incluindo carregamentos de dados em massa, antes de iniciar a atualização.

No início da atualização, o Neptune gera um snapshot com um nome composto de `preupgrade` seguido por um identificador gerado automaticamente com base nas informações do cluster de banco de dados. Você não será cobrado por esse snapshot e poderá usá-lo para restaurar o cluster de banco de dados se algo der errado durante o processo de atualização.

Quando a atualização do mecanismo em si for concluída, a nova versão do mecanismo estará disponível brevemente no sistema operacional antigo, mas em menos de cinco minutos todas as instâncias do cluster iniciarão simultaneamente uma atualização do sistema operacional. O cluster de banco de dados ficará indisponível nesse momento por cerca de seis minutos. Você poderá retomar as workloads de gravação após a conclusão da atualização.

Esse processo gera os seguintes eventos:

- Mensagens de eventos por cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Mensagens de eventos por instância:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`

- DB instance restarted

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para essa versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Versão de manutenção do Amazon Neptune, versão 1.1.1.0.R2 (16/05/2022)

Desde 16/05/2022, a versão 1.1.1.0.R2 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Important

A atualização para esta versão do mecanismo a partir de uma versão anterior à **`1.1.0.0`** também aciona uma atualização do sistema operacional em todas as instâncias do cluster de banco de dados. Como as solicitações de gravação ativas que ocorrem durante a atualização do sistema operacional não serão processadas, você deve pausar todas as workloads de gravação no cluster que está sendo atualizado, incluindo carregamentos de dados em massa, antes de iniciar a atualização.

Para concluir a atualização com êxito, todas as sub-redes em cada zona de disponibilidade (AZ) devem ter pelo menos um endereço IP disponível por instância do Neptune. Por exemplo, se houver uma instância de gravador e duas instâncias de leitor na sub-rede 1 e duas instâncias de leitor na sub-rede 2, a sub-rede 1 deverá ter pelo menos três endereços IP livres e a sub-rede 2 deverá ter pelo menos dois endereços IP livres antes de iniciar a atualização.

No início da atualização, o Neptune gera um snapshot com um nome composto de `preupgrade` seguido por um identificador gerado automaticamente com base nas informações do cluster de banco de dados. Você não será cobrado por esse snapshot e poderá usá-lo para restaurar o cluster de banco de dados se algo der errado durante o processo de atualização.

Quando a atualização do mecanismo em si for concluída, a nova versão do mecanismo estará disponível brevemente no sistema operacional antigo, mas em menos de cinco minutos todas as instâncias do cluster iniciarão simultaneamente uma atualização do sistema operacional. O cluster de banco de dados ficará indisponível nesse momento por alguns minutos. Você poderá retomar as workloads de gravação após a conclusão da atualização.

Esse processo gera os seguintes eventos:

- Mensagens de eventos por cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`

- Mensagens de eventos por instância:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

> **Note**
>
> Há uma alteração significativa nesta versão para o código que usa o openCypher com autenticação do IAM. Até o momento, a string do host na assinatura do IAM incluía o protocolo, como `bolt://`, por exemplo:
>
> ```
> "Host":"bolt://(host URL):(port)"
> ```
>
> A partir da versão do mecanismo `1.1.1.0`, o protocolo deve ser omitido:
>
> ```
> "Host":"(host URL):(port)"
> ```
>
> Consulte Usar o protocolo Bolt para ver exemplos.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.1.1.0.R2, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Versão compatível mais antiga do Gremlin: `3.5.2`
- Versão compatível mais recente do Gremlin: `3.5.4`
- openCypher versão: `Neptune-9.0.20190305-1.0`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.1.1.0.R2 do mecanismo

O cluster será atualizado com esta versão de patch de manutenção automaticamente durante a janela de manutenção seguinte se você estiver executando a versão `1.1.1.0` do mecanismo.

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão.

Atualizar para esta versão

Important

A atualização para esta versão do mecanismo a partir de qualquer versão anterior à **`1.1.0.0`** também aciona uma atualização do sistema operacional em todas as instâncias no cluster de banco de dados. Como as solicitações de gravação ativas que ocorrem durante a atualização do sistema operacional não serão processadas, você deve pausar todas as workloads de gravação no cluster que está sendo atualizado, incluindo carregamentos de dados em massa, antes de iniciar a atualização.

No início da atualização, o Neptune gera um snapshot com um nome composto de `preupgrade` seguido por um identificador gerado automaticamente com base nas informações do cluster de banco de dados. Você não será cobrado por esse snapshot e poderá usá-lo para restaurar o cluster de banco de dados se algo der errado durante o processo de atualização.

Quando a atualização do mecanismo em si for concluída, a nova versão do mecanismo estará disponível brevemente no sistema operacional antigo, mas em menos de cinco minutos todas as instâncias do cluster iniciarão simultaneamente uma atualização do sistema operacional. O cluster de banco de dados ficará indisponível nesse momento por cerca de seis minutos. Você poderá retomar as workloads de gravação após a conclusão da atualização.

Esse processo gera os seguintes eventos:

- Mensagens de eventos por cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Mensagens de eventos por instância:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para essa versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.1.0.0 (19/11/2021)

Desde 19/11/2021, a versão 1.1.0.0 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Important

A atualização para esta versão do mecanismo a partir de uma versão anterior à **`1.1.0.0`** também aciona uma atualização do sistema operacional em todas as instâncias do cluster de banco de dados. Como as solicitações de gravação ativas que ocorrem durante a atualização do sistema operacional não serão processadas, você deve pausar todas as workloads de gravação no cluster que está sendo atualizado, incluindo carregamentos de dados em massa, antes de iniciar a atualização.

Para concluir a atualização com êxito, todas as sub-redes em cada zona de disponibilidade (AZ) devem ter pelo menos um endereço IP disponível por instância do Neptune. Por exemplo, se houver uma instância de gravador e duas instâncias de leitor na sub-rede 1 e duas instâncias de leitor na sub-rede 2, a sub-rede 1 deverá ter pelo menos três endereços IP livres e a sub-rede 2 deverá ter pelo menos dois endereços IP livres antes de iniciar a atualização.

No início da atualização, o Neptune gera um snapshot com um nome composto de `preupgrade` seguido por um identificador gerado automaticamente com base nas informações do cluster de banco de dados. Você não será cobrado por esse snapshot e poderá usá-lo para restaurar o cluster de banco de dados se algo der errado durante o processo de atualização.

Quando a atualização do mecanismo em si for concluída, a nova versão do mecanismo estará disponível brevemente no sistema operacional antigo, mas em menos de cinco minutos todas as instâncias do cluster iniciarão simultaneamente uma atualização do sistema operacional. O cluster de banco de dados ficará indisponível nesse momento por alguns minutos. Você poderá retomar as workloads de gravação após a conclusão da atualização. Esse processo gera os seguintes eventos:

- Mensagens de eventos por cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Mensagens de eventos por instância:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

> **Note**
>
> A partir desta versão do mecanismo, o Neptune não é mais compatível com os tipos de instância R4. Se você estiver usando uma instância R4 no cluster de banco de dados, deverá substituí-la manualmente por um tipo de instância diferente antes de fazer a atualização para esta versão. Se a instância de gravador for uma R4, siga estas instruções para movê-la.

Versões de patch subsequentes para esta versão

- Versão de manutenção: 1.1.0.0.R2 (16/05/2022)
- Versão de manutenção: 1.1.0.0.R3 (23/12/2022)

Novos recursos nesta versão do mecanismo

- Introdução de instâncias de banco de dados T4g de uso geral e R6g otimizadas para memória, equipadas com o processador AWSGraviton2. As instâncias baseadas no Graviton2 oferecem uma relação preço/desempenho significativamente melhor do que as instâncias comparáveis baseadas em x86 da geração atual para uma série de workloads. As aplicações funcionam normalmente nesses novos tipos de instância e não há necessidade de portar o código da aplicação ao realizar a atualização para eles.

  Para obter mais informações sobre disponibilidade de regiões e preços, consulte a página de preços do Amazon Neptune.
- Introdução de modelos personalizados no Neptune ML
- Adição de suporte para consultas de inferência do SPARQL no Neptune ML.
- Adição de um novo endpoint de fluxos para dados de grafos de propriedades, a saber:

  ```
  https://Neptune-DNS:8182/propertygraph/stream
  ```

  O formato de saída desse endpoint, denominado `PG_JSON`, é exatamente o mesmo da saída de formato `GREMLIN_JSON` do antigo `gremlin/stream`.

  O novo endpoint `propertygraph/stream` estende o suporte aos fluxos do Neptune para o openCypher e substitui o endpoint `gremlin/stream` pelo formato de saída `GREMLIN_JSON`associado.

Melhorias nesta versão do mecanismo

- Fizemos melhorias nos fluxos do Neptune:
  - Adição de um campo `commitTimestamp` ao objeto `records`, o formato de resposta do log de alterações dos fluxos do Neptune, para fornecer um carimbo de data e hora para cada registro em um fluxo de logs de alterações.
  - Adição de um valor `LATEST` ao parâmetro `iteratorType`, permitindo que você recupere o último eventId válido dos fluxos. Consulte Chamar a API do Streams.
- Adição de suporte para obter a pontuação de confiança da inferência nas consultas de classificação e regressão de nós do Gremlin.
- Adição de suporte para a cláusula `OPTIONAL MATCH` no openCypher.
- Adição de suporte para a cláusula `MERGE` no openCypher.
- Adição de suporte para usar `ORDER BY` em cláusulas `WITH` no openCypher.
- Adição de suporte para compreensão de padrões ao openCypher e suporte estendido para expressão de padrões além da verificação de existência.
- Suporte estendido para as cláusulas `DELETE` e `DELETE DETACH` no openCypher, para que agora elas possam ser usadas com outras cláusulas de atualização.
- Suporte estendido para as cláusulas `CREATE` e `UPDATE` usadas com `RETURN` no openCypher.
- Adição de suporte ao mecanismo do DFE para as etapas `limit`, `range` e `skip` do Gremlin.
- Melhoria da execução de consultas no mecanismo do DFE quando `explain` nem `profile` é solicitado.
- Melhoria da execução de consultas no mecanismo do DFE para a expressão `value`.
- Melhoria de vários padrões de inserção condicional do Gremlin encadeados para evitar exceções de modificação simultânea e permitir o encadeamento de padrões de consulta como estes:
  - Inserção condicional de vértices por ID, como:

```
g.V(ID).fold().coalesce(unfold(), g.addV("L1").property(id,ID))
```

  - Inserção condicional de vértices com vários rótulos, como:

```
g.V(ID).fold().coalesce(unfold(), g.addV("L1::L2").property(id,ID))
```

  - Inserção condicional de bordas por ID, como:

```
g.E(ID).fold().coalesce(unfold(), V(from).addE(label).to(V(to)).property(id, ID))
```

- Inserção condicional de bordas com vários rótulos, como:

```
g.E(ID).fold().coalesce(unfold(),
 g.addE(label).from(V(from)).to(V(to)).property(id, ID))
```

- Inserção condicional seguida por uma consulta, como:

```
g.V(ID).fold().coalesce(unfold(),
 g.addV("L1").property(id,ID)).project("myvalues").by(valueMap())
```

- Inserção condicional com propriedades adicionadas, como:

```
g.V(ID).fold().coalesce(unfold(),
 g.addV("L1").property(id,ID).property("name","pumba"))
```

Defeitos corrigidos nesta versão do mecanismo

- Desativação do atributo estatísticas em tipos de instância `T3.medium`, que não podiam aceitá-lo.
- Correção de um erro do SPARQL em `explain` com uma função `IN` que usava valores não constantes.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.1.0.0, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.11`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.1.0.0 do mecanismo

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão.

A atualização para esta versão não será automática.

Atualizar para esta versão

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para essa versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Em vez de `--apply-immediately`, é possível especificar `--no-apply-immediately`. Para realizar uma atualização de versão principal, é necessário usar o parâmetro allow-major-version-upgrade. Além disso, não se esqueça de incluir a versão do mecanismo ou ele poderá ser atualizado para outra versão.

Se o cluster usar um grupo de parâmetros de cluster personalizado, não se esqueça de incluir este parâmetro para especificá-lo:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Da mesma forma, se alguma instância no cluster usar um grupo de parâmetros de banco de dados personalizado, não se esqueça de incluir este parâmetro para especificá-lo:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```

Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Versão de manutenção do Amazon Neptune, versão 1.1.0.0.R3 (23/12/2022)

Desde 23/12/2022, a versão 1.1.0.0.R3 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Important

A atualização para esta versão do mecanismo a partir de uma versão anterior à **`1.1.0.0`** também aciona uma atualização do sistema operacional em todas as instâncias do cluster de banco de dados. Como as solicitações de gravação ativas que ocorrem durante a atualização do sistema operacional não serão processadas, você deve pausar todas as workloads de gravação no cluster que está sendo atualizado, incluindo carregamentos de dados em massa, antes de iniciar a atualização.

Para concluir a atualização com êxito, todas as sub-redes em cada zona de disponibilidade (AZ) devem ter pelo menos um endereço IP disponível por instância do Neptune. Por exemplo, se houver uma instância de gravador e duas instâncias de leitor na sub-rede 1 e duas instâncias de leitor na sub-rede 2, a sub-rede 1 deverá ter pelo menos três endereços IP livres e a sub-rede 2 deverá ter pelo menos dois endereços IP livres antes de iniciar a atualização.

No início da atualização, o Neptune gera um snapshot com um nome composto de `preupgrade` seguido por um identificador gerado automaticamente com base nas informações do cluster de banco de dados. Você não será cobrado por esse snapshot e poderá usá-lo para restaurar o cluster de banco de dados se algo der errado durante o processo de atualização.

Quando a atualização do mecanismo em si for concluída, a nova versão do mecanismo estará disponível brevemente no sistema operacional antigo, mas em menos de cinco minutos todas as instâncias do cluster iniciarão simultaneamente uma atualização do sistema

operacional. O cluster de banco de dados ficará indisponível nesse momento por alguns minutos. Você poderá retomar as workloads de gravação após a conclusão da atualização. Esse processo gera os seguintes eventos:

- Mensagens de eventos por cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- Mensagens de eventos por instância:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Melhorias nesta versão do mecanismo

- Melhorias no desempenho e correções para vários operadores do Gremlin, incluindo `repeat`, `coalesce`, `store` e `aggregate`.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um problema de pico de CPU.
- Correção de um erro do openCypher em que as consultas geravam uma string, `"null"`, em vez de um valor nulo em Bolt e SPARQL-JSON.
- Correção de um erro no log de auditoria que fazia com que informações desnecessárias fossem registradas e determinados campos ficassem ausentes nos logs.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.1.0.0.R3, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.11`
- openCypher versão: `Neptune-9.0.20190305-1.0`

- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.1.0.0.R3 do mecanismo

O cluster será atualizado com esta versão de patch de manutenção automaticamente durante a janela de manutenção seguinte se você estiver executando a versão `1.1.0.0` do mecanismo.

Important

A atualização para esta versão do mecanismo a partir de qualquer versão anterior à **`1.1.0.0`** também aciona uma atualização do sistema operacional em todas as instâncias no cluster de banco de dados. Como as solicitações de gravação ativas que ocorrem durante a atualização do sistema operacional não serão processadas, você deve pausar todas as workloads de gravação no cluster que está sendo atualizado, incluindo carregamentos de dados em massa, antes de iniciar a atualização.
No início da atualização, o Neptune gera um snapshot com um nome composto de `preupgrade` seguido por um identificador gerado automaticamente com base nas informações do cluster de banco de dados. Você não será cobrado por esse snapshot e poderá usá-lo para restaurar o cluster de banco de dados se algo der errado durante o processo de atualização.
Quando a atualização do mecanismo em si for concluída, a nova versão do mecanismo estará disponível brevemente no sistema operacional antigo, mas em menos de cinco minutos todas as instâncias do cluster iniciarão simultaneamente uma atualização do sistema operacional. O cluster de banco de dados ficará indisponível nesse momento por cerca de seis minutos. Você poderá retomar as workloads de gravação após a conclusão da atualização.
Esse processo gera os seguintes eventos:

- Mensagens de eventos por cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Mensagens de eventos por instância:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`

- DB instance restarted

> **Note**
>
> A partir desta versão do mecanismo, o Neptune não é mais compatível com os tipos de instância R4. Se você estiver usando uma instância R4 no cluster de banco de dados, deverá substituí-la manualmente por um tipo de instância diferente antes de fazer a atualização para esta versão. Se a instância de gravador for uma R4, siga estas instruções para movê-la.

Atualizar para esta versão

O Amazon Neptune 1.1.0.0.R3 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.0.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.0.0 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```

Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Versão de manutenção do Amazon Neptune, versão 1.1.0.0.R2 (16/05/2022)

Desde 16/05/2022, a versão 1.1.0.0.R2 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Important

A atualização para esta versão do mecanismo a partir de uma versão anterior à **`1.1.0.0`** também aciona uma atualização do sistema operacional em todas as instâncias do cluster de banco de dados. Como as solicitações de gravação ativas que ocorrem durante a atualização do sistema operacional não serão processadas, você deve pausar todas as workloads de gravação no cluster que está sendo atualizado, incluindo carregamentos de dados em massa, antes de iniciar a atualização.
No início da atualização, o Neptune gera um snapshot com um nome composto de `preupgrade` seguido por um identificador gerado automaticamente com base nas informações do cluster de banco de dados. Você não será cobrado por esse snapshot e poderá usá-lo para restaurar o cluster de banco de dados se algo der errado durante o processo de atualização.
Quando a atualização do mecanismo em si for concluída, a nova versão do mecanismo estará disponível brevemente no sistema operacional antigo, mas em menos de cinco minutos todas as instâncias do cluster iniciarão simultaneamente uma atualização do sistema operacional. O cluster de banco de dados ficará indisponível nesse momento por alguns minutos. Você poderá retomar as workloads de gravação após a conclusão da atualização.
Esse processo gera os seguintes eventos:

- Mensagens de eventos por cluster:

    - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
    - `Database cluster major version has been upgraded`
  - Mensagens de eventos por instância:
    - `Applying off-line patches to DB instance`
    - `DB instance shutdown`
    - `Finished applying off-line patches to DB instance`
    - `DB instance restarted`

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro em que o cache interno de credenciais não estava sendo limpo adequadamente para endpoints que não eram de consulta, como o endpoint de status.
- Correção de um erro que fazia com que o atraso na replicação aumentasse após uma atualização do mecanismo.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.1.0.0.R2, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.11`
- openCypher versão: `Neptune-9.0.20190305-1.0`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.1.0.0.R2 do mecanismo

O cluster será atualizado com esta versão de patch de manutenção automaticamente durante a janela de manutenção seguinte se você estiver executando a versão `1.1.0.0` do mecanismo.

> **Important**
>
> A atualização para esta versão do mecanismo a partir de qualquer versão anterior à **`1.1.0.0`** também aciona uma atualização do sistema operacional em todas as instâncias no

cluster de banco de dados. Como as solicitações de gravação ativas que ocorrem durante a atualização do sistema operacional não serão processadas, você deve pausar todas as workloads de gravação no cluster que está sendo atualizado, incluindo carregamentos de dados em massa, antes de iniciar a atualização.
No início da atualização, o Neptune gera um snapshot com um nome composto de `preupgrade` seguido por um identificador gerado automaticamente com base nas informações do cluster de banco de dados. Você não será cobrado por esse snapshot e poderá usá-lo para restaurar o cluster de banco de dados se algo der errado durante o processo de atualização.
Quando a atualização do mecanismo em si for concluída, a nova versão do mecanismo estará disponível brevemente no sistema operacional antigo, mas em menos de cinco minutos todas as instâncias do cluster iniciarão simultaneamente uma atualização do sistema operacional. O cluster de banco de dados ficará indisponível nesse momento por cerca de seis minutos. Você poderá retomar as workloads de gravação após a conclusão da atualização.
Esse processo gera os seguintes eventos:

- Mensagens de eventos por cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Mensagens de eventos por instância:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Note

A partir desta versão do mecanismo, o Neptune não é mais compatível com os tipos de instância R4. Se você estiver usando uma instância R4 no cluster de banco de dados, deverá substituí-la manualmente por um tipo de instância diferente antes de fazer a atualização para esta versão. Se a instância de gravador for uma R4, siga estas instruções para movê-la.

Atualizar para esta versão

O Amazon Neptune 1.1.0.0.R2 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.0.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.0.0 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.0.5.1 (01/10/2021)

Desde 01/10/2021, a versão 1.0.5.1 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Versões de patch subsequentes para esta versão

- Versão: 1.0.5.1.R2 (26/10/2021)
- Versão: 1.0.5.1.R3 (13/01/2022)
- Versão de manutenção: 1.0.5.1.R4 (16/05/2022)

Novos recursos nesta versão do mecanismo

- Adição de um cache de resultados para armazenar em cache os resultados das consultas especificadas.
- Adição de suporte de data/hora ao openCypher no Neptune.
- Adição de suporte para acesso de `List` e `Map` a elementos ao openCypher no Neptune.

Melhorias nesta versão do mecanismo

- Os nomes dos endpoints do openCypher no Neptune deixaram de diferenciar maiúsculas de minúsculas.
- Melhoria do explain do openCypher.
- Melhoria dos padrões de consulta de upsert único do Gremlin que terminam com as etapas `iterate()` e `profile()`.
- Melhoria do desempenho das funções `keys()` e `property()` do Gremlin.
- A etapa `dedup()` do Gremlin é executada no DFE quando usada com escopo global.
- Os seguintes predicados `HAS` do Gremlin são executados no mecanismo do DFE quando o mecanismo do DFE está habilitado:
  - `EQ`
  - `NEQ`
  - `LT`
  - `LTE`
  - `GT`
  - `GTE`
  - `BETWEEN`
  - `INSIDE`

  - `OUTSIDE`
  - `WITHIN`
  - `AND (connectives)`
  - `OR (connectives)`
- Melhoria do desempenho de consultas LIMIT.
- Melhoria do desempenho das consultas gerais de agregação do openCypher.

Defeitos corrigidos nesta versão do mecanismo

- Correção do erro do Gremlin que permitia que uma borda fosse conectada a outra borda.
- Correção do erro do Gremlin que fazia com que uma estratégia de junção abaixo do ideal fosse escolhida.
- Correção do erro do Gremlin que fazia com que a serialização de nós e relacionamentos parasse quando mais de cem propriedades estavam presentes.
- Correção de um erro que retardava o planejamento da execução de consultas com padrões de grafos grandes.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.5.1, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.11`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.5.1 do mecanismo

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão.

Você não atualizará automaticamente para esta versão.

Atualizar para esta versão

O Amazon Neptune 1.0.5.1 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.1 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.1 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Versão de manutenção do Amazon Neptune, versão 1.0.5.1.R4 (16/05/2022)

Desde 16/05/2022, a versão 1.0.5.1.R4 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.5.1.R4, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.11`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.5.1.R4 do mecanismo

O cluster será atualizado com esta versão de patch de manutenção automaticamente durante a janela de manutenção seguinte se você estiver executando a versão `1.0.5.1` do mecanismo.

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão.

Atualizar para esta versão

O Amazon Neptune 1.0.5.1.R4 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.1 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.1 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.0.5.1.R3 (13/01/2022)

Desde 13/01/2022, a versão 1.0.5.1.R3 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro que pode causar um vazamento de recursos quando uma consulta não consegue adquirir todos os recursos necessários.
- Correção de um pequeno vazamento de memória durante a execução da consulta causado por uma alocação de memória não reivindicada.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.5.1.R3, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.11`

- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.5.1.R3 do mecanismo

O cluster será atualizado com essa versão de patch automaticamente durante a próxima janela de manutenção se você estiver executando a versão `1.0.5.1` do mecanismo.

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão.

Atualizar para esta versão

O Amazon Neptune 1.0.5.1.R3 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para essa versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.1 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.1 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```

Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.0.5.1.R2 (26/10/2021)

Desde 26/10/2021, a versão 1.0.5.1.R2 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro que causava a reinicialização do servidor quando ocorria um erro transitório ao criar uma versão mais antiga de um elemento de grafo, sob isolamento de leitura repetível. Em vez disso, o Neptune gera um erro para que o cliente possa tentar novamente.
- Correção de um erro que causava a reinicialização do servidor quando ocorria um erro transitório durante uma única atualização de cardinalidade. Em vez disso, o Neptune gera um erro para que o cliente possa tentar novamente.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.5.1.R2, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.11`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.5.1.R2 do mecanismo

O cluster será atualizado com essa versão de patch automaticamente durante a próxima janela de manutenção se você estiver executando a versão `1.0.5.1` do mecanismo.

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão.

Atualizar para esta versão

O Amazon Neptune 1.0.5.1.R2 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para essa versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.1 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.1 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.0.5.0 (27/07/2021)

Desde 27/07/2021, a versão 1.0.5.0 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Versões de patch subsequentes para esta versão

- Versão: 1.0.5.0.R2 (16/08/2021)
- Versão: 1.0.5.0.R3 (15/09/2021)
- Versão de manutenção: 1.0.5.0.R5 (16/05/2022)

Novos recursos nesta versão do mecanismo

- O Neptune ML foi lançado para uso em produção com muitos atributos novos e não está mais no modo de laboratório.
- Adição de suporte inicial para a linguagem de consulta openCypher, no modo de laboratório. O openCypher é o padrão de código aberto para a linguagem de consulta Cypher. Sua sintaxe é especificada na Cypher Query Language Reference (versão 9) e é mantida pelo projeto openCypher.

  Consulte Acessar o grafo do Neptune com o openCypher para obter informações sobre a implementação da linguagem do Neptune.

  O suporte para o protocolo Bolt, que os clientes do Neptune usam para consultas do openCypher, também é compatível. Consulte Usar o protocolo Bolt para fazer consultas do openCypher ao Neptune.

  O suporte para openCypher agora é habilitado automaticamente, mas depende do Mecanismo DFE do Neptune, que o momento só está disponível no modo de laboratório. A configuração `DFEQueryEngine` padrão no parâmetro de cluster de banco de dados `neptune_lab_mode` agora é `DFEQueryEngine=viaQueryHint`, o que significa que o mecanismo está habilitado, mas é usado apenas para consultas que tenham a dica de consulta `useDFE` presente e definida como `true`. Se você desabilitar o mecanismo do DFE por meio da configuração`DFEQueryEngine=disabled`, não poderá usar o openCypher.
- Adição de suporte para o protocolo HTTP SPARQL 1.1 Graph Store. Consulte Usar o protocolo HTTP do SPARQL 1.1 Graph Store (GSP) no Amazon Neptune.
- Alteração da configuração padrão do modo de laboratório do Mecanismo DFE do Neptune para `viaQueryHint`, o que significa que o mecanismo do DFE é habilitado por padrão, mas é usado apenas para consultas que tenham a dica de consulta `useDFE` presente e definida como `true`.

- Adição de uma nova métrica do Amazon CloudWatch, `StatsNumStatementsScanned`, para monitorar o cálculo de estatísticas para o mecanismo do DFE do Neptune. Consulte Usando a `StatsNumStatementsScanned` CloudWatch métrica para monitorar o cálculo estatístico.

Melhorias nesta versão do mecanismo

- Adição de suporte para o TinkerPop 3.4.11.

> **Important**
>
> Foi feita uma alteração na versão 3.4.11 do TinkerPop que melhora a exatidão de como as consultas são processadas, mas, no momento, às vezes pode afetar gravemente o desempenho das consultas.
> Por exemplo, uma consulta desse tipo pode apresentar uma lentidão significativa:
>
> ```
> g.V().hasLabel('airport').
>   order().
>     by(out().count(),desc).
>   limit(10).
>   out()
> ```
>
> Os vértices após a etapa limite agora são buscados de uma forma não ideal devido à alteração do TinkerPop 3.4.11. Para evitar isso, é possível modificar a consulta adicionando a etapa barrier() a qualquer momento após `order().by()`. Por exemplo:
>
> ```
> g.V().hasLabel('airport').
>   order().
>     by(out().count(),desc).
>   limit(10).
>   barrier().
>   out()
> ```

- A dica de consulta `joinOrder` do SPARQL agora é compatível com o mecanismo de consulta alternativo DFE do Neptune.
- A saída da API de status do Neptune foi expandida e reorganizada para oferecer maior clareza sobre as configurações e os atributos do cluster de banco de dados.

A nova saída tem um objeto `features` de nível superior que contém informações de status sobre os atributos do cluster de banco de dados e um objeto `settings` de nível superior que contém informações de configurações. Para avaliar o novo formato, consulte Exemplo da saída do comando instance status.

- O tratamento dos logs de alterações de streaming foi aprimorado quando os fluxos `AFTER_SEQUENCE_NUMBER` são solicitados com o último ID de evento no servidor, quando esse ID de evento já expirou. O servidor não vai mais gerar um erro de ID de evento expirado se o ID de evento solicitado for o removido mais recentemente no servidor.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro do Gremlin relacionado à ordenação dos valores numéricos.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.5.0, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.11`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.5.0 do mecanismo

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão.

Você não atualizará automaticamente para esta versão.

Atualizar para esta versão

O Amazon Neptune 1.0.5.0 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.0 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Versão de manutenção do Amazon Neptune, versão 1.0.5.0.R5 (16/05/2022)

Desde 16/05/2022, a versão 1.0.5.0.R5 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.5.0.R5, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.11`

- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.5.0.R5 do mecanismo

O cluster será atualizado com esta versão de patch de manutenção automaticamente durante a janela de manutenção seguinte se você estiver executando a versão 1.0.5.0 do mecanismo.

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão.

Atualizar para esta versão

O Amazon Neptune 1.0.5.0.R5 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para essa versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.0 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```

Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.0.5.0.R3 (15/09/2021)

Desde 15/09/2021, a versão 1.0.5.0.R3 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro que fazia com que o mecanismo parasse de responder em qualquer uma destas situações:
  - Um carregamento em massa acontece ao mesmo tempo em que o cálculo automático de estatísticas está ocorrendo.
  - Um cálculo de estatísticas foi solicitado manualmente ao mesmo tempo em que um já estava ocorrendo.
- Correção de um erro na detecção de um deadlock e na aquisição de bloqueios que poderiam causar a falha do mecanismo.
- Correção de um erro do Gremlin em que o mecanismo gerava um erro ao encontrar dados desconhecidos de um endpoint de ML remoto em uma consulta de inferência do Gremlin.
- Correção de vários erros nas APIs de gerenciamento de modelos de ML relacionados a trabalhos de transformação de modelos e recomendações de instâncias.
- Correção de um erro que poderia fazer com que um mecanismo travasse durante a geração de IDs de nós e bordas.
- Correção de um erro que retardava a geração de planos para consultas com padrões de grafos grandes.
- Correção de um erro do openCypher que podia fazer com que uma consulta parasse ao recuperar um nó com mais de cem propriedades.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.5.0.R3, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.11`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.5.0.R3 do mecanismo

O cluster será atualizado com esta versão de patch automaticamente durante a janela de manutenção seguinte se você estiver executando a versão 1.0.5.0 do mecanismo.

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão.

Atualizar para esta versão

O Amazon Neptune 1.0.5.0.R3 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.0 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
> ```

```
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.0.5.0.R2 (16/08/2021)

Desde 16/08/2021, a versão 1.0.5.0.R2 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Defeitos corrigidos nesta versão do mecanismo

- Desativação de uma otimização realizada na versão do mecanismo `1.0.5.0` que fazia com que o cache de pesquisa do Neptune sobrevivesse às reinicializações do mecanismo em réplicas. As reinicializações das réplicas agora limpam o cache de pesquisa.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.5.0.R2, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.11`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.5.0.R2 do mecanismo

O cluster será atualizado com essa versão de patch automaticamente durante a próxima janela de manutenção se você estiver executando a versão `1.0.5.0` do mecanismo.

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão.

Atualizar para esta versão

O Amazon Neptune 1.0.5.0.R2 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.0 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.0.4.2 (01/06/2021)

> **Note**
>
> A versão 1.0.4.2.R2 do mecanismo foi a primeira versão da 1.0.4.2 realmente a ser lançada.

Tópicos

Mecanismo do Amazon Neptune versão 1.0.4.2.R5 (16/08/2021)

Desde 16/08/2021, a versão 1.0.4.2.R5 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Defeitos corrigidos nesta versão do mecanismo

- Desativação de uma otimização realizada na versão do mecanismo `1.0.4.2.R4` que fazia com que o cache de pesquisa do Neptune sobrevivesse às reinicializações do mecanismo em réplicas. As reinicializações das réplicas agora limpam o cache de pesquisa.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.4.2.R5, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.10`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.4.2.R5 do mecanismo

O cluster será atualizado com essa versão de patch automaticamente durante a próxima janela de manutenção se você estiver executando a versão `1.0.4.2` do mecanismo.

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão.

Mecanismo do Amazon Neptune versão 1.0.4.2.R4 (23/07/2021)

Desde 23/07/2021, a versão 1.0.4.2.R4 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Melhorias nesta versão do mecanismo

- Melhoria do comportamento do cache de pesquisa para evitar a limpeza redundante do cache após a execução da redefinição rápida em uma réplica.
- Melhoria do tratamento dos logs de alterações de streaming quando os fluxos `AFTER_SEQUENCE_NUMBER` são solicitados com o último ID de evento no servidor, quando esse ID de evento já expirou. O servidor não vai mais gerar um erro de ID de evento expirado se o ID de evento solicitado for o removido mais recentemente no servidor.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro introduzido na versão 1.0.4.0.R1 em que as consultas não exibiam a totalidade dos valores de string maiores que 760 caracteres. Os termos afetados por esse erro eram literais do RDF e URIs, ou IDs do Gremlin, chaves e valores de string.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.4.2.R4, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.10`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.4.2.R4 do mecanismo

O cluster será atualizado com essa versão de patch automaticamente durante a próxima janela de manutenção se você estiver executando a versão `1.0.4.2` do mecanismo.

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão.

Mecanismo do Amazon Neptune versão 1.0.4.2.R3 (28/06/2021)

Desde 28/06/2021, a versão 1.0.4.2.R3 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Problemas conhecidos nesta versão do mecanismo

Problema:

Um erro do SPARQL de não respeitar o tipo de mídia em um cabeçalho `Accept` na presença de espaços.

Por exemplo, uma consulta com `-H "Accept: text/csv; q=1.0, */*; q=0.1"` exibe uma saída JSON em vez de uma saída CSV.

Solução:

Se você remover os espaços na cláusula `Accept` no cabeçalho, o mecanismo exibirá a saída no formato correto solicitado. Em outras palavras, em vez de `-H "Accept: text/csv; q=1.0, */*; q=0.1"` , use:

```
-H "Accept: text/csv;q=1.0,*/*;q=0.1"
```

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro na limpeza do cache de pesquisa em réplicas após uma redefinição rápida.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.4.2.R3, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.10`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.4.2.R3 do mecanismo

Esta versão de patch é opcional, a menos que o cluster de banco de dados esteja usando uma ou mais instâncias `R5d`. Se o cluster tiver instâncias `R5d`, ele será atualizado automaticamente na janela de manutenção seguinte. Caso contrário, ele não será atualizado automaticamente para esta versão de patch.

É possível atualizar a `1.0.4.2.R2` para esta versão `1.0.4.2.R3` manualmente usando o comando apply-pending-maintenance-action da AWS CLI (a API ApplyPendingMaintenanceAction).

Mecanismo do Amazon Neptune versão 1.0.4.2.R2 (01/06/2021)

Desde 01/06/2021, a versão 1.0.4.2.R2 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Versões de patch subsequentes para esta versão

- Versão: 1.0.4.2.R3 (28/06/2021)

Problemas conhecidos nesta versão do mecanismo

Problema:

Um erro do SPARQL de não respeitar o tipo de mídia em um cabeçalho `Accept` na presença de espaços.

Por exemplo, uma consulta com `-H "Accept: text/csv; q=1.0, */*; q=0.1"` exibe uma saída JSON em vez de uma saída CSV.

Solução:

Se você remover os espaços na cláusula `Accept` no cabeçalho, o mecanismo exibirá a saída no formato correto solicitado. Em outras palavras, em vez de `-H "Accept: text/csv; q=1.0, */*; q=0.1"`, use:

```
-H "Accept: text/csv;q=1.0,*/*;q=0.1"
```

Novos recursos nesta versão do mecanismo

- Adição do novo tipo de instância R5d, que inclui um cache de pesquisa para acelerar as leituras em casos de uso que envolvam um alto volume de valor de propriedade ou pesquisas literais de RDF. Consulte O cache de pesquisa do Neptune pode acelerar as consultas de leitura.
- Adição de um novo parâmetro de modo de laboratório que permite que o mecanismo experimental do DFE seja invocado somente por consulta com a dica de consulta `useDFE`.

Melhorias nesta versão do mecanismo

- Adição de suporte para o TinkerPop 3.4.10.
- Adição de suporte para usar a etapa de configuração `withStrategies( )` ao enviar solicitações de script do Gremlin. Especificamente, as `SubgraphStrategy`, `PartitionStrategy`, `ReadOnlyStrategy`, `EdgeLabelVerificationStrategy` e `ReservedKeysVerificationStrategy` são todas compatíveis.
- Adição da otimização de percursos `V()` no meio de uma consulta. Anteriormente, esses percursos não eram otimizados no Neptune.
- Adição de suporte para URNs RFC 2141 para serem usados como parâmetros `baseUri` e `namedGraphUri` para um carregamento em massa.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro do Gremlin no analisador em que consultas incorretas eram tratadas como válidas.
- Correção de um erro do Gremlin em que desdobrar um efeito colateral `aggregate()` com `cap().unfold()` a um `valueMap()` lançava uma exceção.
- Correção de um erro do Gremlin em que algumas etapas `property()` após uma etapa `addV()` falhavam com o erro “não é possível converter em string”.
- Correção de um erro do Gremlin para evitar que alguns padrões de inserção condicional lançassem exceções de modificação simultânea.
- Correção de um erro do Gremlin para que o tempo limite da solicitação de consulta agora não exceda o tempo limite da sessão.

- Correção de um erro do SPARQL em que as atualizações usando LOAD ou UNLOAD podiam falhar com um código HTTP 500 em vez do código HTTP 400 quando o servidor remoto não estava disponível.
- Correção de um erro em que as chamadas da API do fluxo falhavam quando os valores `commitNum` ou `opNum` maiores que o limite de números inteiros assinados de 32 bits (2.147.483.647) eram usados.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.4.2.R2, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.10`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.4.2.R2 do mecanismo

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão.

Você não atualizará automaticamente para esta versão.

Atualizar para esta versão

O Amazon Neptune 1.0.4.2.R2 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para essa versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.2 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.2 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.0.4.2.R1 (27/05/2021)

A versão 1.0.4.2.R1 do mecanismo nunca foi implantada.

Mecanismo do Amazon Neptune versão 1.0.4.1 (08/12/2020)

Desde 08/12/2020, a versão 1.0.4.1 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Versões de patch subsequentes para esta versão

- Versão: 1.0.4.1.R1.1 (22/03/2021)
- Versão: 1.0.4.1.R2 (24/02/2021)

> **Important**
>
> Versão: 1.0.4.0 (12/10/2020) tornou o TLS 1.2 e o HTTPS obrigatórios para todas as conexões com o Amazon Neptune. No entanto, um erro nesta versão permitia que

conexões HTTP e/ou conexões TLS desatualizadas continuassem funcionando para clientes que haviam definido anteriormente um parâmetro de cluster de banco de dados para impedir a imposição de conexões HTTPS.
Esse erro foi corrigido nas versões de patch 1.0.4.0.R2 e 1.0.4.1.R2, mas a correção causava falhas de conexão inesperadas quando os patches eram instalados automaticamente. Por esse motivo, os dois patches foram revertidos e só podem ser instalados manualmente, para que você tenha a chance de atualizar sua configuração para o TLS 1.2.
A necessidade de usar SSL/TLS para todas as conexões com o Neptune afeta as conexões com o console do Gremlin, o driver do Gremlin, Gremlin Python, .NET, nodeJs, APIs REST e também conexões do balanceador de carga. Se você usa HTTP ou uma versão mais antiga do TLS para qualquer uma ou todas, deve atualizar o cliente e os drivers relevantes e alterar o código para usar HTTPS exclusivamente antes de atualizar o sistema para os patches mais recentes.

Novos recursos nesta versão do mecanismo

- Introdução do recurso Neptune ML, que traz recursos avançados de machine learning para o Amazon Neptune. Consulte Amazon Neptune ML para machine learning em grafos.
- Adição de uma operação `UNLOAD` personalizada do SPARQL para remover dados recuperados de uma fonte remota. Consulte SPARQL UPDATE UNLOAD.

Melhorias nesta versão do mecanismo

- Otimização de alguns padrões de inserção condicional do Gremlin para evitar exceções de modificação simultânea.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro do Gremlin que poderia causar a falta de resultados para um padrão específico de consultas que usavam a etapa `as()`.
- Correção de um erro do Gremlin que poderia causar erros ao usar a etapa `project()` aninhada dentro de outra etapa, como `union()`.
- Correção de um erro do Gremlin na etapa `project()`.

- Correção de um erro do Gremlin no percurso baseado em strings em que a etapa `none()` não funcionava.
- Correção de um erro do Gremlin no percurso baseado em strings em que um mapa vazio não era aceito como argumento para a etapa `inject()`.
- Correção de um erro do Gremlin na execução de um percurso baseado em strings no mecanismo do DFE em que um método terminal como o `toList()` não funcionava corretamente.
- Correção de um erro do Gremlin que não fechava as transações que usavam a etapa `iterate()` em consultas de string.
- Correção de um erro do Gremlin que poderia fazer com que consultas que usavam o padrão `is(P.gte(0))` lançassem uma exceção em algumas situações.
- Correção de um erro do Gremlin que poderia fazer com que consultas que usavam o padrão `order().by(T.id)` lançassem uma exceção em algumas situações.
- Correção de um erro do Gremlin que poderia fazer com que consultas que usavam o padrão `addV().aggregate()` produzissem resultados incorretos em algumas situações.
- Correção de um erro do Gremlin que poderia fazer com que consultas que usavam a etapa `path()` seguida pela etapa `project()` lançassem uma exceção em algumas situações.
- Correção de um erro do SPARQL em que a função `SUBSTR` sinaliza um erro em vez de gerar uma string vazia.
- Correção de um erro no mecanismo do DFE que podia fazer com que as operações de junção em planos de consulta sem bloqueio gerassem resultados incorretos na presença de variáveis não vinculadas.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.4.1, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.8`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.4.1 do mecanismo

O cluster será atualizado com essa versão de patch automaticamente durante a próxima janela de manutenção se você estiver executando a versão `1.0.4.1` do mecanismo.

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão.

Atualizar para esta versão

O Amazon Neptune 1.0.4.1 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.1 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.1 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.0.4.1.R1.1 (22/03/2021)

Desde 22/03/2021, a versão 1.0.4.1.R1.1 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Defeitos corrigidos nesta versão do mecanismo

- Desativação de uma otimização para padrões de inserção condicional do Gremlin que podem ser adicionados ou anexados a rótulos e propriedades existentes.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.4.1.R1.1, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.8`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.4.1.R1.1 do mecanismo

O cluster será atualizado com essa versão de patch automaticamente durante a próxima janela de manutenção se você estiver executando a versão `1.0.4.1` do mecanismo.

Atualizar para esta versão

O Amazon Neptune 1.0.4.1.R1.1 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.1 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.1 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.0.4.1.R2 (24/02/2021)

Desde 24/02/2021, a versão 1.0.4.1.R2 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Versões de patch subsequentes para esta versão

- Versão: 1.0.4.1.R2.1 (11/03/2021)

Novos recursos nesta versão do mecanismo

- O Neptune já é compatível com a compactação de arquivos únicos em formato `bzip2` para carregamentos em massa. Consulte Formatos de dados de carga.

Defeitos corrigidos nesta versão do mecanismo

- Correção do erro Versão: 1.0.4.0 (12/10/2020) que permitia conexões com o Neptune usando HTTP ou versões anteriores do TLS, em vez de HTTPS e TLS 1.2.

> **Important**
>
> A necessidade de usar SSL/TLS para todas as conexões com o Neptune pode ser uma mudança significativa. Isso afeta as conexões com o console do Gremlin, o driver do Gremlin, Gremlin Python, .NET, NodeJs, APIs REST e também conexões do balanceador de carga. Se você usa HTTP ou uma versão mais antiga do TLS para qualquer uma ou todas, deve atualizar o cliente e os drivers relevantes antes de instalar esse patch e alterar o código para usar exclusivamente HTTPS.

- Correção do erro do Gremlin em que `InternalFailureException` era definida como o código de resposta em determinadas circunstâncias quando ocorria uma `ConcurrentModificationException`.
- Correção de um erro do Gremlin em que, em determinadas condições, a atualização de bordas ou vértices poderia causar uma `InternalFailureException` transitória.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.4.1.R2, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.8`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.4.1.R2 do mecanismo

O cluster será atualizado com essa versão de patch automaticamente durante a próxima janela de manutenção se você estiver executando a versão `1.0.4.1` do mecanismo.

Atualizar para esta versão

O Amazon Neptune 1.0.4.1.R2 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.1 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.1 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.0.4.1.R2.1 (11/03/2021)

Desde 11/03/2021, a versão 1.0.4.1.R2.1 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Defeitos corrigidos nesta versão do mecanismo

- Desativação de uma otimização para padrões de inserção condicional do Gremlin que podem ser adicionados ou anexados a rótulos e propriedades existentes.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.4.1.R2.1, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.8`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.4.1.R1.2 do mecanismo

O cluster será atualizado com essa versão de patch automaticamente durante a próxima janela de manutenção se você estiver executando a versão `1.0.4.1.R2` do mecanismo.

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão.

Atualizar para esta versão

O Amazon Neptune 1.0.4.1.R2.1 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.1.R2 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
```

```
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.1.R2 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.0.4.0 (12/10/2020)

Desde 12/10/2020, a versão 1.0.4.0 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Versões de patch subsequentes para esta versão

- Versão: 1.0.4.0.R2 (24/02/2021)

Novos recursos nesta versão do mecanismo

- Adição da compactação de nível de quadro para o Gremlin.

Melhorias nesta versão do mecanismo

- O Amazon Neptune agora exige o uso do Secure Sockets Layer (SSL) com o protocolo TLSv1.2 para todas as conexões com o Neptune em todas as regiões, usando estes pacotes de criptografia forte:
  - `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`
  - `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`

- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA`

Isso vale para conexões REST e WebSocket com o Neptune e significa que você deve usar HTTPS em vez de HTTP ao se conectar ao Neptune em todas as regiões.

Como as conexões de clientes usando HTTP ou TLS 1.1 não serão mais aceitas em nenhum lugar, garanta que os clientes e o código tenham sido atualizados para usar TLS 1.2 e HTTPS antes de atualizar para esta versão do mecanismo.

> **Important**
>
> A necessidade de usar SSL/TLS para todas as conexões com o Neptune pode ser uma mudança significativa. Isso afeta as conexões com o console do Gremlin, o driver do Gremlin, Gremlin Python, .NET, NodeJs, APIs REST e também conexões do balanceador de carga. Se você estiver usando HTTP para qualquer um ou todos eles, agora deverá atualizar o cliente e os drivers relevantes e alterar o código para usar HTTPS ou as conexões falharão.
>
> Um erro nesta versão permitia que conexões HTTP e/ou conexões TLS desatualizadas continuassem funcionando para clientes que haviam definido anteriormente um parâmetro de cluster de banco de dados para impedir a imposição de conexões HTTPS. Esse erro foi corrigido nas versões de patch 1.0.4.0.R2 e 1.0.4.1.R2, mas a correção causava falhas de conexão inesperadas quando os patches eram instalados automaticamente.
>
> Por esse motivo, os dois patches foram revertidos e só podem ser instalados manualmente, para que você tenha a chance de atualizar sua configuração para o TLS 1.2.

- Atualização do TinkerPop para a versão 3.4.8. Esta é uma atualização compatível com versões anteriores. Consulte o log de alterações do TinkerPop para ver as novidades.
- Melhoria do desempenho da etapa `properties()` do Gremlin.
- Adição de detalhes sobre `BindOp` e `MultiplexerOp` nos relatórios de explain e profile.
- Adição da pré-busca de dados para melhorar o desempenho quando há falhas de cache.

- Adição de uma nova configuração `allowEmptyStrings` no parâmetro `parserConfiguration` do carregador em massa que permite que strings vazias sejam tratadas como valores de propriedades válidos em carregamentos de CSV (consulte Parâmetros de solicitação do carregador do Neptune).
- O carregador agora permite um ponto e vírgula com escape em colunas CSV de vários valores.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um possível vazamento de memória do Gremlin relacionado à etapa `both()`.
- Correção de um erro em que as métricas de solicitação estavam ausentes porque um endpoint terminado com "/" não estava sendo tratado corretamente.
- Correção de um erro que fazia com que as réplicas ficassem atrasadas e reiniciassem sob carga pesada quando o mecanismo do DFE estava habilitado no modo de laboratório.
- Correção de um erro que impedia que a mensagem de erro correta fosse relatada quando um carregamento em massa falhava devido a uma condição de falta de memória.
- Correção de um erro do SPARQL em que a codificação de caracteres era colocada no cabeçalho Content-Encoding nas respostas de consultas do SPARQL. Em vez disso, agora `charset` é colocado no cabeçalho Content-Type, permitindo que os clientes HTTP reconheçam o conjunto de caracteres que está sendo usado automaticamente.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.4.0, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.8`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.3.0 do mecanismo

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão.

Você não atualizará automaticamente para esta versão.

Atualizar para esta versão

O Amazon Neptune 1.0.4.0 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.0 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.0.4.0.R2 (24/02/2021)

Desde 24/02/2021, a versão 1.0.4.0.R2 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Defeitos corrigidos nesta versão do mecanismo

- Correção do erro Versão: 1.0.4.0 (12/10/2020) que permitia conexões com o Neptune usando HTTP ou versões anteriores do TLS, em vez de HTTPS e TLS 1.2.

  > **Important**
  >
  > A necessidade de usar SSL/TLS para todas as conexões com o Neptune pode ser uma mudança significativa. Isso afeta as conexões com o console do Gremlin, o driver do Gremlin, Gremlin Python, .NET, NodeJs, APIs REST e também conexões do balanceador de carga. Se você usa HTTP ou uma versão mais antiga do TLS para qualquer uma ou todas, deve atualizar o cliente e os drivers relevantes antes de instalar esse patch e alterar o código para usar exclusivamente HTTPS.

- Correção de um erro no carregamento em massa em CSV envolvendo rótulos que terminam com #.
- Correção do erro do Gremlin em que `InternalFailureException` era definida como o código de resposta em determinadas circunstâncias quando ocorria uma `ConcurrentModificationException`.
- Correção de um erro do Gremlin em que, em determinadas condições, a atualização de bordas ou vértices poderia causar uma `InternalFailureException` transitória.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.4.0.R2, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.8`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.4.0.R2 do mecanismo

O cluster será atualizado com essa versão de patch automaticamente durante a próxima janela de manutenção se você estiver executando a versão `1.0.4.0` do mecanismo.

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão.

Atualizar para esta versão

O Amazon Neptune 1.0.4.0.R2 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.0 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.0.3.0 (03/08/2020)

Desde 03/08/2020, a versão 1.0.3.0 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Versões de patch subsequentes para esta versão

- Versão: 1.0.3.0.R2 (12/10/2020)
- Versão: 1.0.3.0.R3 (19/02/2021)

Novos recursos nesta versão do mecanismo

- O Neptune introduziu um novo mecanismo de consulta alternativo (DFE) que pode acelerar significativamente a execução de consultas. Consulte O mecanismo de consulta alternativo (DFE) do Amazon Neptune.
- O DFE se baseia em estatísticas pré-geradas sobre os dados de grafos do Neptune que são gerenciadas por meio de novos endpoints de estatísticas. Consulte Estatísticas do DFE.
- Agora é possível excluir trabalhos de carregamento em fila da lista de IDs de carregamento gerada pela API Get-Status do carregador definindo o novo parâmetro `includeQueuedLoads` como FALSE. Consulte Parâmetros da solicitação Get-Status do carregador do Neptune.
- O Neptune agora é compatível com cabeçalhos finais para respostas a consultas do SPARQL que poderão conter um código de erro e uma mensagem se uma solicitação falhar depois de começar a exibir partes da resposta. Consulte Cabeçalhos finais HTTP opcionais para respostas SPARQL de várias partes.
- O Neptune agora também permite habilitar a codificação de respostas em partes para consultas do Gremlin. Como no caso do SPARQL, os fragmentos de resposta têm cabeçalhos finais que poderão conter um código de erro e uma mensagem se ocorrer uma falha após a consulta começar a exibir fragmentos de resposta. Consulte Usar cabeçalhos finais HTTP opcionais para habilitar respostas do Gremlin em várias partes.

Melhorias nesta versão do mecanismo

- Agora você pode fornecer o tamanho das solicitações em lote ao ElasticSearch para pesquisas de texto completo no Gremlin.
- Melhoria do uso de memória para consultas SPARQL GROUP BY.
- Adição de um novo otimizador de consultas do Gremlin para remover determinados filtros não vinculados.
- Aumento do tempo máximo que uma conexão WebSocket autenticada com o IAM pode permanecer aberta, de 36 horas para 10 dias.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro em que, se você enviasse um parâmetro de URL não codificado em uma solicitação POST, o Neptune gerava um código de status HTTP de 500 e uma `InternalServerErrorException`. Agora, o Neptune gera um código de status HTTP 400 e uma `BadRequestException`, com a mensagem: `Failure to process the POST request parameters`.
- Correção de um erro do Gremlin em que uma falha na conexão do WebSocket não era relatada corretamente.
- Correção de um erro do Gremlin que envolvia o desaparecimento de sideEffects.
- Correção de um erro do Gremlin em que o parâmetro `batchsize` de pesquisa de texto completo não era aceito corretamente.
- Correção de um erro do Gremlin para lidar com `toV` e `fromV` individualmente para cada direção em `bothE`.
- Correção de um erro do Gremlin que envolve o `Edge pathType` na etapa `hasLabel`.
- Correção de um erro do SPARQL em que a reordenação de junções com vinculações estáticas não estava funcionando corretamente.
- Correção de um erro SPARQL UPDATE LOAD em que um bucket indisponível do Amazon S3 não era relatado corretamente.
- Correção de um erro do SPARQL em que um problema com um nó SERVICE em uma subconsulta não era relatado corretamente.
- Correção de um erro do SPARQL em que as consultas que continham as condições aninhadas FILTER EXISTS ou FILTER NOT EXISTS não estavam sendo avaliadas corretamente.
- Correção de um erro do SPARQL para tratar corretamente vinculações geradas duplicadas ao chamar endpoints do SPARQL Service por meio de consultas de geração.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.3.0, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.3`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.3.0 do mecanismo

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão.

Se o cluster tiver seu parâmetro `AutoMinorVersionUpgrade` definido como `True`, o cluster será atualizado para esta versão do mecanismo automaticamente duas a três semanas após a data desta versão, durante uma janela de manutenção.

Atualizar para esta versão

O Amazon Neptune 1.0.3.0 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.3.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.3.0 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```

Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.0.3.0.R3 (19/02/2021)

Desde 19/02/2021, a versão 1.0.3.0.R3 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro no carregamento em massa em CSV envolvendo rótulos que terminam com `#`.
- Correção de um erro do Gremlin que poderia causar a falta de resultados para um padrão específico de consultas que usam a etapa `as()`.
- Correção de um erro do Gremlin que poderia causar erros ao usar a etapa `project()` aninhada dentro de outra etapa, como `union()`.
- Correção de um erro do Gremlin na execução de percursos de strings no mecanismo experimental do DFE quando um método terminal como `toList()` é usado.
- Correção de um erro do Gremlin que não fechava uma transação ao usar a etapa `iterate()` em uma consulta de string.
- Correção de um erro do Gremlin que poderia fazer com que consultas usando o padrão `is(P.gte(0))` lançassem uma exceção em determinadas condições.
- Correção do erro do Gremlin em que `InternalFailureException` era definida como o código de resposta em determinadas circunstâncias quando ocorria uma `ConcurrentModificationException`.
- Correção de um erro do Gremlin em que, em determinadas condições, a atualização de bordas ou vértices poderia causar uma `InternalFailureException` transitória.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.3.0.R3, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.8`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.3.0.R3 do mecanismo

O cluster será atualizado com essa versão de patch automaticamente durante a próxima janela de manutenção se você estiver executando a versão `1.0.3.0` do mecanismo.

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão.

Atualizar para esta versão

O Amazon Neptune 1.0.3.0.R3 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.3.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.3.0 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias,

ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
```

```
proceeding with the upgrade.
```

Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.0.3.0.R2 (12/10/2020)

Desde 12/10/2020, a versão 1.0.3.0.R2 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Melhorias nesta versão do mecanismo

- Melhoria do desempenho da etapa `properties()` do Gremlin.
- Adição de detalhes sobre `Bind0p` e `MultiplexerOp` nos relatórios de explain e profile.
- Para respostas de consulta do SPARQL, adicionamos `charset` ao cabeçalho Content-Type, permitindo que os clientes HTTP reconheçam o conjunto de caracteres que está sendo usado automaticamente.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro do SPARQL em que `CancellationException` não era tratada.
- Correção de um erro do SPARQL em que as consultas com opções aninhadas não funcionavam corretamente.
- Correção de um erro do SPARQL em LOAD em que `ConcurrentModificationException` poderia fazer com que uma consulta travasse.
- Correção de um erro do SPARQL que impedia que as respostas das consultas fossem compactadas com gzip.
- Correção de um erro do Gremlin na etapa `groupBy()`.
- Correção de um erro do Gremlin relacionado ao uso de uma etapa `aggregate()` dentro de uma etapa `local()`.

- Correção de um erro do Gremlin relacionado ao uso de `bothE()` seguido por um predicado que usa valores agregados.
- Correção de um erro do Gremlin relacionado ao uso da etapa `bothE()` com a etapa `repeat()`.
- Correção de um possível vazamento de memória do Gremlin relacionado à etapa `both()`.
- Correção de um erro em que as métricas de solicitação estavam ausentes porque um endpoint terminado com "/" não estava sendo tratado corretamente.
- Correção de um erro que poderia lançar uma `ThrottlingException` mesmo quando a fila de solicitações não estivesse cheia.
- Correção de um erro na busca do status de carregamento quando um carregamento falhava por um motivo, como `LOAD_DATA_FAILED_DUE_TO_FEED_MODIFIED_OR_DELETE`.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.3.0.R2, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.3`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.3.0.R2 do mecanismo

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão.

Se o cluster tiver seu parâmetro `AutoMinorVersionUpgrade` definido como `True`, o cluster será atualizado para esta versão do mecanismo automaticamente duas a três semanas após a data desta versão, durante uma janela de manutenção.

Atualizar para esta versão

O Amazon Neptune 1.0.3.0.R2 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.3.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.3.0 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot

manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.0.2.2 (09/03/2020)

Desde 09/03/2020, a versão 1.0.2.2 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Versões de patch subsequentes para esta versão

- Versão: 1.0.2.2.R2 (02/04/2020)
- Versão: 1.0.2.2.R3 (22/07/2020)
- Versão: 1.0.2.2.R4 (23/07/2020)
- Versão: 1.0.2.2.R5 (12/10/2020)
- Versão: 1.0.2.2.R6 (19/02/2021)

Melhorias nesta versão do mecanismo

- Foram adicionadas informações à API de status sobre transações que estão sendo revertidas. Consulte Status de instância.
- Foi atualizada a versão do Apache TinkerPop para 3.4.3.

  A versão 3.4.3 é compatível com a versão anterior compatível com o Neptune (3.4.1). Ela apresenta uma pequena alteração no comportamento: o Gremlin não retorna mais um erro quando você tenta fechar uma sessão que não existe (consulte Evitar erro ao fechar sessões que não existem).
- Foram removidos gargalos de desempenho na execução de etapas de pesquisa de texto completo do Gremlin.

Defeitos corrigidos nesta versão do mecanismo

- Foi corrigido um bug do SPARQL na manipulação de padrões de gráficos vazios em consultas.
- Foi corrigido um bug do SPARQL na manipulação de pontos-e-vírgulas não codificados em consultas com codificação de URL.
- Foi corrigido um bug do Gremlin na manipulação de vértices repetidos na etapa `Union`.
- Foi corrigido um bug do Gremlin que fazia com que algumas consultas com um `.simplePath()` ou um `.cyclicPath()` dentro de um `.repeat()` retornassem resultados incorretos.
- Foi corrigido um bug do Gremlin que fazia `.project()` retornar resultados incorretos se sua travessia filho não retornasse nenhuma solução.
- Foi corrigido um bug do Gremlin em que erros de conflitos de leitura/gravação geravam um `InternalFailureException`, em vez de um `ConcurrentModificationException`.
- Foi corrigido um bug do Gremlin que causava falhas `.group().by(...).by(values("property"))`.
- Foram corrigidos bugs do Gremlin na saída do perfil para etapas de pesquisa de texto completo.
- Foi corrigido um vazamento de recursos em sessões do Gremlin.
- Foi corrigido um bug que impedia a API de status de relatar a versão correta que pode ser solicitada em alguns casos.
- Correção de um erro do carregador em massa que permitia que um URL para um local diferente do Amazon S3 fosse usado como a origem em uma solicitação de carregamento em massa.
- Foi corrigido um bug do carregador em massa no status detalhado da carga.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.2.2, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.3`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.2.2 do mecanismo

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão.

Se o cluster tiver seu parâmetro `AutoMinorVersionUpgrade` definido como `True`, o cluster será atualizado para esta versão do mecanismo automaticamente duas a três semanas após a data desta versão, durante uma janela de manutenção.

Atualizar para esta versão

O Amazon Neptune 1.0.2.2 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
> ```

```
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.0.2.2.R6 (19/02/2021)

Desde 19/02/2021, a versão 1.0.2.2.R6 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Defeitos corrigidos nesta versão do mecanismo

- Correção do erro do Gremlin em que `InternalFailureException` era definida como o código de resposta em determinadas circunstâncias quando ocorria uma `ConcurrentModificationException`.
- Correção de um erro do Gremlin em que, em determinadas condições, a atualização de bordas ou vértices poderia causar uma `InternalFailureException` transitória.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.2.2.R6, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.8`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.2.2.R6 do mecanismo

O cluster será atualizado com essa versão de patch automaticamente durante a próxima janela de manutenção se você estiver executando a versão `1.0.2.2` do mecanismo.

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão.

Atualizar para esta versão

O Amazon Neptune 1.0.2.2.R6 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.0.2.2.R5 (12/10/2020)

Desde 12/10/2020, a versão 1.0.2.2.R5 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Melhorias nesta versão do mecanismo

- Melhoria do desempenho da etapa `properties()` do Gremlin.
- Adição de detalhes sobre `BindOp` e `MultiplexerOp` nos relatórios de explain e profile.
- Para respostas de consulta do SPARQL, adicionamos `charset` ao cabeçalho Content-Type, permitindo que os clientes HTTP reconheçam o conjunto de caracteres que está sendo usado automaticamente.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro do SPARQL em que `CancellationException` não era tratada.
- Correção de um erro do SPARQL em que as consultas com opções aninhadas não funcionavam corretamente.
- Correção de um erro do SPARQL em LOAD em que `ConcurrentModificationException` poderia fazer com que uma consulta travasse.
- Correção de um erro do SPARQL que impedia que as respostas das consultas fossem compactadas com gzip.
- Correção de um erro do Gremlin na etapa `groupBy()`.
- Correção de um erro do Gremlin relacionado ao uso de uma etapa `aggregate()` dentro de uma etapa `local()`.
- Correção de um erro do Gremlin relacionado ao uso de `bothE()` seguido por um predicado que usa valores agregados.
- Correção de um erro do Gremlin relacionado ao uso da etapa `bothE()` com a etapa `repeat()`.
- Correção de um possível vazamento de memória do Gremlin relacionado à etapa `both()`.
- Correção de um erro em que as métricas de solicitação estavam ausentes porque um endpoint terminado com “/” não estava sendo tratado corretamente.
- Correção de um erro que poderia lançar uma `ThrottlingException` mesmo quando a fila de solicitações não estivesse cheia.

- Correção de um erro na busca do status de carregamento quando um carregamento falhava por um motivo, como `LOAD_DATA_FAILED_DUE_TO_FEED_MODIFIED_OR_DELETE`.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.2.2.R5, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.3`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.2.2.R5 do mecanismo

O cluster será atualizado com essa versão de patch automaticamente durante a próxima janela de manutenção se você estiver executando a versão `1.0.2.2` do mecanismo.

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão.

Atualizar para esta versão

O Amazon Neptune 1.0.2.2.R5 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
```

```
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.0.2.2.R4 (23/07/2020)

Desde 23/07/2020, a versão 1.0.2.2.R4 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Melhorias nesta versão do mecanismo

- Melhoria do uso da memória com a liberação mais frequente de memória não utilizada de volta para o sistema operacional.
- Também foi aprimorado o uso de memória para consultas SPARQL GROUP BY.
- Aumento do tempo máximo que uma conexão WebSocket autenticada com o IAM pode permanecer aberta, de 36 horas para 10 dias.
- Adição de uma métrica `BufferCacheHitRatio` do CloudWatch, que pode ser útil para diagnosticar a latência da consulta e ajustar os tipos de instância. Consulte Métricas do Neptune.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro no fechamento de conexões WebSocket do IAM inativas ou expiradas. O Neptune agora envia um quadro de fechamento antes de fechar a conexão.
- Correção de um erro do SPARQL na avaliação de consultas que contêm condições aninhadas de FILTER EXISTS e/ou FILTER NOT EXISTS.

- Correção de um erro de encerramento de consultas do SPARQL que causava o bloqueio de threads no servidor em determinadas condições extremas.
- Correção de um erro do Gremlin que envolve o Edge PathType na etapa `hasLabel`.
- Correção de um erro do Gremlin para lidar com `toV` e `fromV` individualmente para cada direção em `bothE`.
- Correção de um erro do Gremlin que envolvia o desaparecimento de sideEffects.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.2.2.R4, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.3`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.2.2.R4 do mecanismo

O cluster será atualizado com essa versão de patch automaticamente durante a próxima janela de manutenção se você estiver executando a versão `1.0.2.2` do mecanismo.

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão.

Atualizar para esta versão

O Amazon Neptune 1.0.2.2.R4 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para essa versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
```

```
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.0.2.2.R3 (22/07/2020)

A versão 1.0.2.2.R3 do mecanismo foi incorporada à versão 1.0.2.2.R4 do mecanismo.

Mecanismo do Amazon Neptune versão 1.0.2.2.R2 (02/04/2020)

Desde 02/04/2020, a versão 1.0.2.2.R2 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Melhorias nesta versão do mecanismo

- Agora é possível enfileirar até 64 trabalhos de carregamento em massa, em vez de esperar um trabalho terminar para iniciar o próximo Também é possível executar uma solicitação de carga em fila dependente da conclusão com êxito de um ou mais trabalhos de carga enfileirados anteriormente usando o parâmetro `dependencies` do comando `load`. Consulte Comando do carregador do Neptune.
- A saída da pesquisa de texto completo agora pode ser classificada (consulte Parâmetros de pesquisa de texto completo).

- Agora há um parâmetro do cluster de banco de dados para invocar fluxos do Neptune e o atributo foi movido para fora do modo de laboratório. Consulte Habilitar os fluxos do Neptune.

Defeitos corrigidos nesta versão do mecanismo

- Corrigida uma falha estocástica na inicialização do servidor que atrasava a criação da instância.
- Corrigido um problema do otimizador em que as instruções do `BIND` na consulta inicializavam o otimizador com padrões não seletivos no planejamento da ordem de junção.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.2.2.R2, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.3`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.2.2.R2 do mecanismo

O cluster será atualizado com essa versão de patch automaticamente durante a próxima janela de manutenção se você estiver executando a versão `1.0.2.2` do mecanismo.

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão.

Atualizar para esta versão

O Amazon Neptune 1.0.2.2.R2 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
```

```
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.0.2.1 (22/11/2019)

Versões de patch subsequentes para esta versão

- Versão: 1.0.2.1.R6 (22/04/2020)
- Versão: 1.0.2.1.R5 (22/04/2020) Esta versão de patch não foi implantada.
- Versão: 1.0.2.1.R4 (20/12/2019)
- Versão: 1.0.2.1.R3 (12/12/2019)
- Versão: 1.0.2.1.R2 (25/11/2019)

Novos recursos nesta versão do mecanismo

- Adição de recursos de pesquisa de texto completo por meio da integração com o Amazon OpenSearch Service. Consulte Pesquisa de texto completo do Neptune

- Adição da opção que usa o modo de laboratório para criar um quarto índice (um índice OSGP) para grandes números de predicados Consulte Índice OSGP.
- Adição de um modo de detalhes ao Explain do SPARQL. Consulte Usar o SPARQL `explain` e Saída do modo de detalhes para obter mais detalhes.
- Adição de informações do modo de laboratório ao relatório de status do mecanismo. Para mais detalhes, consulte Status de instância.
- Os snapshots de cluster de banco de dados agora podem ser copiados entre regiões da AWS. Consulte Cópia de um snapshot.

Melhorias nesta versão do mecanismo

- Melhor desempenho ao lidar com um grande número de predicados.
- Otimização de consulta aprimorada. Embora isso deva ser completamente transparente para os clientes, nós encorajamos você a testar os aplicativos antes de atualizá-los para garantir que eles se comportem conforme o esperado.
- Pequenas melhorias no relatório de erros.
- Adição de otimizações nas etapas `.project()` e `.identity()` do Gremlin.
- Adição de otimizações nos casos não terminais `.union()` do Gremlin.
- Adição de suporte nativo às travessias `.path().by()` do Gremlin.
- Adição de suporte nativo ao `.coalesce()` do Gremlin.
- Otimização adicional de gravação em massa.
- Agora exigimos que as conexões HTTPS utilizem pelo menos a versão TLS 1.2 ou superior para evitar que codificações desatualizadas/não seguras sejam usadas.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro ao lidar com a travessia `addE()` interna do Gremlin.
- Correção de um bug do Gremlin causado pelo vazamento de anotações AST das travessias secundárias para a primária.
- Correção de um bug que ocorria no Gremlin quando o `.otherV()` era chamado após o `select()`.
- Correção de um bug que gerava falha em algumas etapas do `.hasLabel()` caso aparecessem após uma etapa do `bothE()`.

- Pequenas correções do .sum() e .project() do Gremlin.
- Correção de um bug no processamento de consultas SPARQL que não possuíam uma chave de fechamento.
- Correção de alguns bugs pequenos no Explain do SPARQL.
- Correção de um erro ao lidar com solicitações simultâneas para obter o status da carga.
- Redução da memória usada para executar algumas travessias do Gremlin com as etapas do `.project()`.
- Correção de comparações numéricas de valores especiais no SPARQL. Consulte Conformidade com padrões.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.2.1, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.1`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.2.1 do mecanismo

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão.

Você não atualizará automaticamente para esta versão.

Atualizar para esta versão

O Amazon Neptune 1.0.2.1 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para essa versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
```

```
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot

manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.0.2.1.R6 (22/04/2020)

Desde 22/04/2020, a versão 1.0.2.1.R6 do mecanismo está sendo implantada de forma geral. Observe que leva vários dias para que uma nova versão fique disponível em todas as regiões.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um bug em que `ConcurrentModificationConflictException` e `TransactionException` não eram convertidos em um `NeptuneGremlinException`, fazendo com que `InternalFailureException` fosse retornado aos clientes.
- Correção de um erro em que o Neptune relatava seu status como íntegro antes que o servidor estivesse completamente pronto.
- Correção de um erro em que as confirmações de dicionário de transação do usuário ficavam fora de ordem quando dois mapeamentos `value->id` estavam sendo inseridos simultaneamente.

- Correção de um bug na serialização de status de carregamento.
- Correção de um bug de sessões do Gremlin.
- Correção de um erro em que o Neptune não conseguia lançar uma exceção quando o servidor falhava ao iniciar.
- Correção de um erro em que o Neptune não conseguia enviar um quadro de fechamento do WebSocket antes de fechar o canal.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.2.1.R6, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.1`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.2.1.R6 do mecanismo

O cluster será atualizado com essa versão de patch automaticamente durante a próxima janela de manutenção se você estiver executando a versão `1.0.2.1` do mecanismo.

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão.

Atualizar para esta versão

O Amazon Neptune 1.0.2.1.R6 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para esta versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
```

```
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome

que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.0.2.1.R5 (22/04/2020)

A versão 1.0.2.1.R5 do mecanismo nunca foi implantada.

Mecanismo do Amazon Neptune versão 1.0.2.1.R4 (20/12/2019)

Melhorias nesta versão do mecanismo

- Agora, o Neptune sempre tenta colocar qualquer chamada de pesquisa de texto completo primeiro no pipeline de execução. Isso reduz o volume de chamadas para o OpenSearch, o que pode melhorar significativamente o desempenho. Consulte Execução de consulta de pesquisa de texto completo.
- Agora, o Neptune lançará uma `IllegalArgumentException` se você tentar acessar um vértice, uma borda ou uma propriedade inexistente. Antes, o Neptune lançava uma `UnsupportedOperationException` nessa situação.

Por exemplo, se tentar adicionar uma borda fazendo referência a um vértice inexistente, agora você gerará um `IllegalArgumentException`.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um bug do Gremlin em que uma travessia `union` dentro de um `project-by` não retorna resultados ou retorna resultados incorretos.
- Correção de um bug do Gremlin que fazia com que as etapas `.project().by()` aninhadas retornassem resultados incorretos.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.2.1.R4, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.1`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.2.1.R4 do mecanismo

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão.

No entanto, a atualização automática para esta versão não é compatível.

Atualizar para esta versão

O Amazon Neptune 1.0.2.1.R4 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para essa versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
```

```
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome

que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.0.2.1.R3 (12/12/2019)

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro em que o índice `OSGP` era desabilitado mesmo que o recurso estivesse habilitado corretamente ao ser usado em Modo de laboratório com o valor `ObjectIndex` no parâmetro `neptune_lab_mode`.
- Correção de um bug que afetava as consultas do Gremlin com um `.fold()` dentro de uma etapa do `.project().by()`. Por exemplo, gerou resultados incompletos para a consulta:

  ```
  g.V().project("a").by(valueMap().fold())
  ```

- Correção de um afunilamento de desempenho em carregamentos em massa de dados RDF.
- Correção de um erro que gerava falha em réplicas quando os fluxos eram habilitados e a réplica era reiniciada antes da principal.

- Correção de um bug no qual os certificados SSL alternados nas instâncias não eram coletados sem reiniciar a instância.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.2.1.R3, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.1`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.2.1.R3 do mecanismo

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão.

No entanto, a atualização automática para esta versão não é compatível.

Atualizar para esta versão

O Amazon Neptune 1.0.2.1.R3 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para essa versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
```

```
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.0.2.1.R2 (25/11/2019)

Defeitos corrigidos nesta versão do mecanismo

- Correção de um bug que afetava todas as consultas `project().by()` com travessias de repetição não alternada e travessias não `path()`.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.2.1.R2, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.1`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.2.1.R2 do mecanismo

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão.

No entanto, a atualização automática para esta versão não é compatível.

Atualizar para esta versão

O Amazon Neptune 1.0.2.1.R2 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para essa versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.0.2.0 (08/11/2019)

IMPORTANTE: ESSA VERSÃO DO MECANISMO ESTÁ OBSOLETA

A partir de 19/05/2020, nenhuma nova instância que use esta versão do mecanismo será criada.

Esta versão do mecanismo foi substituída pela versão 1.0.2.1, que contém todas as correções de erros desta versão, além de atributos adicionais, como a integração de pesquisa em texto completo, o suporte ao índice OSGP e a cópia de cluster de snapshot de banco de dados entre regiões da AWS.

A partir de 1.º de junho de 2020, o Neptune atualizará automaticamente todos os clusters que estiverem executando essa versão do mecanismo para o patch mais recente da versão 1.0.2.1 durante a janela de manutenção seguinte. Você pode atualizar manualmente antes disso, conforme descrito aqui.

Se você tiver qualquer problema com a atualização, entre em contato conosco por meio do AWS Support ou dos AWSfóruns de desenvolvedores da .

Versões de patch subsequentes para esta versão

- Versão: 1.0.2.0.R3 (05/05/2020)
- Versão: 1.0.2.0.R2 (21/11/2019)

Novos recursos nesta versão do mecanismo

Além das atualizações de manutenção, esta versão adiciona uma nova funcionalidade para oferecer suporte para mais de uma versão do mecanismo por vez (consulte Manter o cluster de banco de dados do Amazon Neptune).

Como resultado, a numeração das versões do mecanismo foi alterada (consulte Numeração de versão antes da versão 1.3.0.0 do mecanismo).

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.2.0, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.1`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.2.0 do mecanismo

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão.

Você não atualizará automaticamente para esta versão.

Atualizar para esta versão

O Amazon Neptune 1.0.2.0 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para essa versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.0 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```

Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.0.2.0.R3 (05/052020)

IMPORTANTE: ESSA VERSÃO DO MECANISMO ESTÁ OBSOLETA

A partir de 19/05/2020, nenhuma nova instância que use esta versão do mecanismo será criada.

Esta versão do mecanismo foi substituída pela versão 1.0.2.1, que contém todas as correções de erros desta versão, além de atributos adicionais, como a integração de pesquisa em texto completo, o suporte ao índice OSGP e a cópia de cluster de snapshot de banco de dados entre regiões da AWS.

A partir de 1.º de junho de 2020, o Neptune atualizará automaticamente todos os clusters que estiverem executando essa versão do mecanismo para o patch mais recente da versão 1.0.2.1 durante a janela de manutenção seguinte. Você pode atualizar manualmente antes disso, conforme descrito aqui.

Se você tiver qualquer problema com a atualização, entre em contato conosco por meio do AWS Support ou dos AWSfóruns de desenvolvedores da .

Defeitos corrigidos nesta versão do mecanismo

- Correção de um bug em que `ConcurrentModificationConflictException` e `TransactionException` eram relatados como `InternalFailureException` genéricos.
- Correção de erros nas verificações de integridade que causavam reinicializações frequentes do servidor durante a inicialização.
- Correção de um erro em que os dados não eram visíveis nas réplicas porque as confirmações estavam fora de ordem sob determinadas condições.
- Correção de um erro na serialização de status de carregamento em que um carregamento falhava devido à falta de permissões de acesso do Amazon S3.

- Foi corrigido um vazamento de recursos em sessões do Gremlin.
- Correção de um erro na verificação de integridade que ocultava o status não íntegro na inicialização dos componentes que gerenciam a autenticação do IAM.
- Correção de um erro em que o Neptune não conseguia enviar um quadro de fechamento do WebSocket antes de fechar o canal.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.2.0.R3, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.1`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.2.0.R3 do mecanismo

O cluster será atualizado com essa versão de patch automaticamente durante a próxima janela de manutenção se você estiver executando a versão `1.0.2.0` do mecanismo.

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para essa versão.

Atualizar para esta versão

O Amazon Neptune 1.0.2.0.R3 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para essa versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.0 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.0.2.0.R2 (21/11/2019)

IMPORTANTE: ESSA VERSÃO DO MECANISMO ESTÁ OBSOLETA

A partir de 19/05/2020, nenhuma nova instância que use esta versão do mecanismo será criada.

Esta versão do mecanismo foi substituída pela versão 1.0.2.1, que contém todas as correções de erros desta versão, além de atributos adicionais, como a integração de pesquisa em texto completo, o suporte ao índice OSGP e a cópia de cluster de snapshot de banco de dados entre regiões da AWS.

A partir de 1.º de junho de 2020, o Neptune atualizará automaticamente todos os clusters que estiverem executando essa versão do mecanismo para o patch mais recente da versão 1.0.2.1 durante a janela de manutenção seguinte. Você pode atualizar manualmente antes disso, conforme descrito aqui.

Se você tiver qualquer problema com a atualização, entre em contato conosco por meio do AWS Support ou dos AWSfóruns de desenvolvedores da .

Defeitos corrigidos nesta versão do mecanismo

- Melhoria da estratégia de cache para páginas sujas no servidor, para que `FreeableMemory` se recupere mais rápido quando o servidor entrar em um estado de memória insuficiente.
- Correção de um bug que podia gerar uma condição de corrida e falha quando várias solicitações de status de carga e/ou de inicialização de carga eram processadas simultaneamente no servidor.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.2.0.R2, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.1`
- SPARQL versão: `1.1`

Caminhos de atualização para a versão 1.0.2.0.R2 do mecanismo

É possível atualizar manualmente qualquer versão anterior do mecanismo do Neptune para esta versão.

No entanto, a atualização automática para esta versão não é compatível.

Atualizar para esta versão

O Amazon Neptune 1.0.2.0.R2 já está disponível ao público em geral.

Se um cluster de banco de dados estiver executando uma versão do mecanismo a partir da qual haja um caminho de atualização para essa versão, ele estará elegível para ser atualizado agora. Você pode atualizar qualquer cluster elegível usando as operações do cluster de banco de dados no console ou usando o SDK. O seguinte comando da CLI atualizará imediatamente um cluster elegível:

Para Linux, OS X ou Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.0 ^
    --apply-immediately
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados nessas instâncias, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso do cluster de banco de dados.

Sempre teste antes de fazer a atualização

Quando uma nova versão principal ou secundária do mecanismo do Neptune for lançada, sempre teste as aplicações do Neptune antes de atualizá-la. Mesmo uma atualização secundária pode introduzir novos atributos ou comportamentos que afetem o código.

Comece comparando as páginas de notas da versão atual com as da versão de destino para ver se haverá alterações nas versões da linguagem de consulta ou outras alterações importantes.

A melhor maneira de testar uma nova versão antes de atualizar o cluster de banco de dados de produção é clonar o cluster de produção para que o clone execute a nova versão do mecanismo. Depois, você pode executar consultas no clone sem afetar o cluster de banco de dados de produção.

Sempre crie um snapshot manual antes de fazer a atualização

Antes de fazer uma atualização, é altamente recomendável sempre criar um snapshot manual do cluster de banco de dados. Ter um snapshot automático só oferece proteção de curto prazo, enquanto um snapshot manual permanece disponível até que você o exclua explicitamente.

Em determinados casos, o Neptune cria um snapshot manual para você como parte do processo de atualização, mas não confie nisso e, em qualquer caso, crie o próprio snapshot manual.

Quando você tiver certeza de que não precisará reverter o cluster de banco de dados para o estado de pré-atualização, poderá excluir explicitamente o snapshot manual criado, bem como o snapshot manual que o Neptune tenha criado. Se o Neptune criar um snapshot manual, ele terá um nome que começa com `preupgrade`, seguido pelo nome do cluster de banco de dados, a versão do mecanismo de origem, a versão do mecanismo de destino e a data.

> **Note**
>
> Se você estiver tentando atualizar com uma ação pendente em andamento, poderá encontrar um erro como o seguinte:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Se você encontrar esse erro, aguarde a conclusão da ação pendente ou acione imediatamente uma janela de manutenção para permitir que a atualização anterior seja concluída.

Para obter mais informações sobre como atualizar a versão do mecanismo, consulte Manter o cluster de banco de dados do Amazon Neptune. Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Mecanismo do Amazon Neptune versão 1.0.1.2 (10/06/2020)

IMPORTANTE: ESTA VERSÃO DO MECANISMO ESTÁ OBSOLETA

A partir de 27/04/2021, nenhuma nova instância que use esta versão do mecanismo será criada.

Melhorias nesta versão do mecanismo

- Agora, o Neptune lançará uma `IllegalArgumentException` se você tentar acessar um vértice, uma borda ou uma propriedade inexistente. Antes, o Neptune lançava uma `UnsupportedOperationException` nessa situação.

  Por exemplo, se tentar adicionar uma borda fazendo referência a um vértice inexistente, agora você gerará um `IllegalArgumentException`.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro em que as confirmações de dicionário de transação do usuário ficavam fora de ordem quando dois mapeamentos `value->id` estavam sendo inseridos simultaneamente.
- Correção de um bug na serialização de status de carregamento.
- Corrigida uma falha estocástica na inicialização do servidor que atrasava a criação da instância.
- Foi corrigido um vazamento de cursor.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.1.2, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.4.1`
- SPARQL versão: `1.1`

Mecanismo do Amazon Neptune versão 1.0.1.1 (26/06/2020)

IMPORTANTE: ESTA VERSÃO DO MECANISMO ESTÁ OBSOLETA

A partir de 27/04/2021, nenhuma nova instância que use esta versão do mecanismo será criada.

Defeitos corrigidos nesta versão do mecanismo

- Corrigido um erro em que as confirmações ficavam fora de ordem quando inseridas simultaneamente.
- Correção de um bug na serialização de status de carregamento.
- Corrigida uma falha estocástica na inicialização do servidor que atrasava a criação da instância.
- Corrigido um vazamento de memória.

Versões de linguagem de consulta compatíveis com esta versão

Antes de atualizar um cluster de banco de dados para a versão 1.0.1.1, assegure-se de que o projeto seja compatível com estas versões da linguagem de consulta:

- Gremlin versão: `3.3.2`
- SPARQL versão: `1.1`

Mecanismo do Amazon Neptune versão 1.0.1.0 (02/07/2019)

IMPORTANTE: ESSA VERSÃO DO MECANISMO ESTÁ OBSOLETA

A partir de 27/04/2021, nenhuma nova instância que use essa versão do mecanismo será criada.

Atualizações do mecanismo do Amazon Neptune de 31/10/2019

Versão: 1.0.1.0.200502.0

IMPORTANTE: ESSA VERSÃO DO MECANISMO ESTÁ OBSOLETA

A partir de 27/04/2021, nenhuma nova instância que use essa versão do mecanismo será criada.

Defeitos corrigidos nesta versão do mecanismo

- Foi corrigido um bug do Gremlin na serialização da resposta da etapa `tree()` quando os clientes se conectam ao Neptune usando `traversal().withRemote(...)` (em outras palavras, usando o código de bytes GLV).

  Esta versão corrige um problema em que os clientes que se conectavam ao Neptune usando `traversal().withRemote(...)` recebiam uma resposta inválida às consultas do Gremlin que continham uma etapa `tree()`.
- Foi corrigido um bug do SPARQL em consultas `DELETE WHERE LIMIT`, em que o processo de encerramento da consulta travaria devido a uma condição de disputa, fazendo com que a consulta expirasse.

Atualizações do mecanismo do Amazon Neptune de 15/10/2019

Versão: 1.0.1.0.200463.0

IMPORTANTE: ESSA VERSÃO DO MECANISMO ESTÁ OBSOLETA

A partir de 27/04/2021, nenhuma nova instância que use essa versão do mecanismo será criada.

Novos recursos nesta versão do mecanismo

- Adição de um recurso Explain/Profile do Gremlin (consulte Analisar a execução de consulta do Neptune usando o `explain` do Gremlin).
- Adição de Suporte para sessões baseadas em script do Gremlin para habilitar a execução de várias travessias do Gremlin em uma única transação.
- Adição de suporte para a extensão de consulta federada do SPARQL no Neptune (consulte SPARQL 1.1 Federated Query e Consultas federadas do SPARQL no Neptune usando a extensão `SERVICE`).
- Adição de um recurso que permite injetar seu próprio `queryId` em uma consulta do Gremlin ou do SPARQL, seja por meio de um parâmetro HTTP URL ou por meio de uma dica de consulta de `queryId` do SPARQL (consulte Injetar um ID personalizado em uma consulta do Gremlin ou do SPARQL no Neptune).
- Adição de um atributo Modo de laboratório ao Neptune que pode permitir que você teste os próximos atributos que ainda não estão prontos para uso na produção.
- Adição de um recurso Fluxos do Neptune futuro que registra em log com segurança todas as alterações feitas no banco de dados em um fluxo que persiste por uma semana. Esse recurso está disponível somente no modo de laboratório.
- Atualizada a semântica formal para transações simultâneas (consulte Semântica de transação no Neptune). Esse recurso fornece garantias padrão do setor em relação à simultaneidade.

  Por padrão, essas semânticas de transação estão habilitadas. Em alguns cenários, esse recurso pode alterar o comportamento atual da carga e reduzir o desempenho da carga. Você pode usar o parâmetro `neptune_lab_mode` do cluster de banco de dados para reverter para a semântica anterior incluindo `ReadWriteConflictDetection=disabled` no valor do parâmetro.

Melhorias nesta versão do mecanismo

- A API Status de instância foi aprimorada relatando qual versão do TinkerPop e qual versão do SPARQL o mecanismo está usando.
- O desempenho do operador de subgráfico do Gremlin foi aprimorado.
- O desempenho da serialização de respostas do Gremlin foi aprimorado.
- O desempenho na etapa Gremlin Union foi aprimorado.
- A latência de consultas simples do SPARQL foi aprimorada.

Defeitos corrigidos nesta versão do mecanismo

- Corrigido um bug do Gremlin em que o tempo limite estava sendo retornado incorretamente como uma falha interna.
- Corrigido um bug do SPARQL em que ORDER BY em um conjunto parcial de variáveis causava um erro interno de servidor.

Atualizações do mecanismo do Amazon Neptune de 19/09/2019

IMPORTANTE: ESSA VERSÃO DO MECANISMO ESTÁ OBSOLETA

A partir de 27/04/2021, nenhuma nova instância que use essa versão do mecanismo será criada.

Versão: 1.0.1.0.200457.0

O Amazon Neptune 1.0.1.0.200457.0 está disponível ao público em geral. Todos os novos clusters de banco de dados do Neptune, incluindo aqueles restaurados de snapshots, serão criados no Neptune 1.0.1.0.200457.0 depois que a atualização do mecanismo for concluída para essa região.

Clusters existentes podem ser atualizados para essa versão imediatamente usando as operações de cluster do banco de dados no console ou usando o SDK. É possível usar o seguinte comando da CLI para atualizar um cluster de banco de dados:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados em todas as instâncias de um cluster de banco de dados, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso dos clusters ou do cluster de banco de dados. É possível visualizar ou alterar as configurações da janela de manutenção no Console do Neptune.

Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Defeitos corrigidos nesta versão do mecanismo

- Corrigido um problema de exatidão do Gremlin introduzido na versão anterior do mecanismo (1.0.1.0.200369.0) removendo a melhoria de desempenho para tratamento de predicados conjuntivos que o causou.
- Corrigido um bug do SPARQL que fazia com que consultas com `DISTINCT` e um único padrão encapsulado em `OPTIONAL` gerassem um `InternalServerError`.

Atualizações do mecanismo do Amazon Neptune de 13/08/2019

IMPORTANTE: ESSA VERSÃO DO MECANISMO ESTÁ OBSOLETA

A partir de 27/04/2021, nenhuma nova instância que use essa versão do mecanismo será criada.

Novos recursos nesta versão do mecanismo

- Adição de uma opção `OVERSUBSCRIBE` ao parâmetro `parallelism` do Comando do carregador do Neptune, o que faz com que o carregador em massa do Neptune use todos os threads e recursos disponíveis.

Melhorias nesta versão do mecanismo

- Melhor desempenho de filtros SPARQL contendo expressões lógicas OR simples.
- Melhor desempenho do Gremlin no tratamento de predicados conjuntivos.

Defeitos corrigidos nesta versão do mecanismo

- Correção de um erro do SPARQL que impedia a subtração de `xsd:duration` de `xsd:date`.
- Corrigido um bug do SPARQL que causava resultados incompletos de alinhamento estático na presença de um `UNION`.
- Corrigido um bug do SPARQL no cancelamento de consulta.
- Corrigido um bug do Gremlin que causava estouro durante a promoção do tipo.
- Corrigido um bug do Gremlin no tratamento de elementos de vértice em etapas de `addE().from().to()`.
- Corrigido um bug do Gremlin (lançado em 26/07/2019 na versão 1.0.1.0.200366.0 do mecanismo) envolvendo o tratamento de duplos e flutuantes NaN em inserções de cardinalidade única.

- Corrigido um bug na geração de planos de consulta envolvendo pesquisas baseadas em propriedades.

Atualizações do mecanismo do Amazon Neptune de 26/07/2019

Versão: 1.0.1.0.200366.0

IMPORTANTE: ESSA VERSÃO DO MECANISMO ESTÁ OBSOLETA

A partir de 27/04/2021, nenhuma nova instância que use essa versão do mecanismo será criada.

Novos recursos nesta versão do mecanismo

- Atualizado para TinkerPop 3.4.1 (consulte Informações de atualização do TinkerPop e Log de alterações do TinkerPop 3.4.1).

  Para clientes do Neptune, estas alterações oferecem novas funcionalidades e melhorias, como:

  - `GraphBinary` está agora disponível como um formato de serialização.
  - Um bug de keep-alive que causava vazamentos de memória no driver TinkerPop Java foi corrigido; portanto, uma solução alternativa não é mais necessária.

  No entanto, em alguns casos, eles podem afetar o código do Gremlin existente no Neptune. Por exemplo:

  - O `valueMap()` retorna agora um `Map<Object,Object>` em vez de um `Map<String,Object>`.
  - O comportamento inconsistente da etapa do `within()` foi corrigido para que funcionasse de forma consistente com outras etapas. Anteriormente, os tipos tinham que corresponder para que as comparações funcionassem. Agora, números de tipos diferentes podem ser comparados com precisão. Por exemplo, `33` agora compara como igual a `33L`, que não comparava anteriormente.
  - Um bug no `ReducingBarrierStep` foi corrigido; portanto, ele agora não retorna nenhum valor se nenhum elemento estiver disponível para a saída.
  - A ordem dos escopos `select()` foi alterada (a ordem agora é `maps`, `side-effects`, `paths`). Isso altera os resultados das consultas raras que combinam `side-effects` e `select` com o mesmo nome de chave para `side-effects` e para `select`.
  - O `bulkSet()` agora faz parte do protocolo GraphSON. As consultas que terminam com `toBulkSet()` não funcionam com clientes mais antigos.
  - Uma parametrização da etapa `Submit()` foi removida do cliente 3.4.

Muitas outras alterações introduzidas no TinkerPop 3.4 não afetam o comportamento atual do Neptune. Por exemplo, o `io()` do Gremlin foi adicionado como uma etapa ao `Traversal` e agora está obsoleto no `Graph`, mas nunca foi habilitado no Neptune.

- Adicionado suporte para propriedades de vértice de única cardinalidade carregador em massa para o Gremlin, para carregar dados de gráfico de propriedade.
- Adicionada uma opção para substituir os valores existentes para uma propriedade de única cardinalidade no carregador em massa.
- Adicionada a capacidade de para recuperar o status de uma consulta do Gremlin e de cancelar uma consulta do Gremlin.
- Adicionada uma dica de consulta para limites de tempo de consulta SPARQL.
- Adicionada a capacidade de ver a função de instância na API de status (consulte Status de instância).
- Adicionado suporte para clonagem de banco de dados (consulte Clonagem de banco de dados no Neptune).

Melhorias nesta versão do mecanismo

- Melhoria da Explicação de consultas SPARQL para mostrar variáveis de gráfico a partir das cláusulas FROM.
- Melhoria do desempenho para SPARQL em filtros, filtros iguais, cláusulas VALUES e contagens de intervalo.
- Melhorado o desempenho de ordenação de etapas do Gremlin.
- Melhorado o desempenho para travessias `.repeat.dedup` do Gremlin.
- Melhoria do desempenho de travessias `valueMap()` e `path().by()` do Gremlin.

Defeitos corrigidos nesta versão do mecanismo

- Corrigidos vários problemas com caminhos de propriedade SPARQL incluindo a operação com gráficos nomeados.
- Corrigido um problema com consultas CONSTRUCT SPARQL causando problemas de memória.
- Corrigido um problema com o analisador RDF Turtle e nomes de locais.
- Corrigido um problema nas mensagens de erro exibidas aos usuários.
- Corrigido um problema com os percursos `repeat()...drop()` do Gremlin.

- Corrigido um problema com a etapa `drop()` do Gremlin.
- Correção de um problema com os filtros de rótulos do Gremlin.
- Corrigido um problema com timeout de consultas do Gremlin.

Atualizações do mecanismo do Amazon Neptune de 02/07/2019

IMPORTANTE: ESSA VERSÃO DO MECANISMO ESTÁ OBSOLETA

A partir de 27/04/2021, nenhuma nova instância que use essa versão do mecanismo será criada.

Defeitos corrigidos nesta versão do mecanismo

- Corrigido um erro que fazia com que determinados padrões com um valor e nome de propriedade vinculados não fossem otimizados.

Versões anteriores do mecanismo do Neptune

Tópicos

Atualizações do mecanismo do Amazon Neptune de 12/06/2019

Versão: 1.0.1.0.200310.0

O Amazon Neptune 1.0.1.0.200310.0 está disponível ao público em geral. Todos os novos clusters de banco de dados do Neptune, incluindo aqueles restaurados de snapshots, serão criados no Neptune 1.0.1.0.200310.0 depois que a atualização do mecanismo for concluída para essa região.

Clusters existentes podem ser atualizados para essa versão imediatamente usando as operações de cluster do banco de dados no console ou usando o SDK. É possível usar o seguinte comando da CLI para atualizar um cluster de banco de dados para esta versão imediatamente:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Os clusters de banco de dados do Neptune serão automaticamente atualizados para a versão 1.0.1.0.200310.0 do mecanismo durante as janelas de manutenção do sistema. O cronograma da aplicação das atualizações depende da configuração da região e da janela de manutenção do cluster de banco de dados, bem como do tipo de atualização.

> **Note**
>
> A janela da manutenção da instância não se aplica às atualizações do mecanismo.

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados em todas as instâncias de um cluster de banco de dados, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso dos clusters ou do cluster de banco de dados. É possível visualizar ou alterar as configurações da janela de manutenção no Console do Neptune.

Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Melhorias

- Corrige um erro em que a inserção e o descarte simultâneos de uma borda podem resultar em várias bordas com o mesmo ID.
- Outras correções e melhorias secundárias.

Atualizações do mecanismo do Amazon Neptune de 01/05/2019

Versão: 1.0.1.0.200296.0

O Amazon Neptune 1.0.1.0.200296.0 está disponível ao público em geral. Todos os novos clusters de banco de dados do Neptune, incluindo aqueles restaurados de snapshots, serão criados no Neptune 1.0.1.0.200296.0 depois que a atualização do mecanismo for concluída para essa região.

Clusters existentes podem ser atualizados para essa versão imediatamente usando as operações de cluster do banco de dados no console ou usando o SDK. É possível usar o seguinte comando da CLI para atualizar um cluster de banco de dados para esta versão imediatamente:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Os clusters de banco de dados do Neptune serão automaticamente atualizados para a versão 1.0.1.0.200296.0 do mecanismo durante as janelas de manutenção do sistema. O cronograma da aplicação das atualizações depende da configuração da região e da janela de manutenção do cluster de banco de dados, bem como do tipo de atualização.

> **Note**
>
> A janela da manutenção da instância não se aplica às atualizações do mecanismo.

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados em todas as instâncias de um cluster de banco de dados, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso dos clusters ou do cluster de banco de dados. É possível visualizar ou alterar as configurações da janela de manutenção no Console do Neptune.

Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Melhorias

- O novo atributo `explain` foi adicionado a consultas do SPARQL no Neptune para ajudar você a visualizar o plano de consulta e executar etapas para otimizá-lo, se necessário. Para obter mais informações, consulte `explain` do SPARQL.
- O desempenho e os relatórios do SPARQL foram melhorados de várias maneiras.

- O desempenho e o comportamento do Gremlin foram melhorados de várias maneiras.
- O tempo limite de consultas `drop( )` de execução longa foi melhorado.
- O desempenho de consultas `otherV( )` foi melhorado.
- Foram adicionados dois campos às informações exibidas ao consultar o status de integridade de um cluster ou uma instância de banco de dados do Neptune, ou seja, o número da versão do mecanismo e a hora de início do cluster ou da instância. Consulte Status de instância.
- A API `Get-Status` do carregador do Neptune agora exibe um campo `startTime` que registra quando um trabalho de carregamento é iniciado.
- O comando do carregador agora usa um parâmetro `parallelism` opcional que permite que você restrinja o número de threads que o carregador usa.

Atualizações do mecanismo do Amazon Neptune de 21/01/2019

Versão: 1,0.1.0.200267.0

O Amazon Neptune 1.0.1.0.200267.0 está disponível ao público em geral. Todos os novos clusters de banco de dados do Neptune, incluindo aqueles restaurados de snapshots, serão criados no Neptune 1.0.1.0.200267.0 depois que a atualização do mecanismo for concluída para essa região.

Clusters existentes podem ser atualizados para essa versão imediatamente usando as operações de cluster do banco de dados no console ou usando o SDK. É possível usar o seguinte comando da CLI para atualizar um cluster de banco de dados para esta versão imediatamente:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Os clusters de banco de dados do Neptune serão automaticamente atualizados para a versão 1.0.1.0.200267.0 do mecanismo durante as janelas de manutenção do sistema. O cronograma da aplicação das atualizações depende da configuração da região e da janela de manutenção do cluster de banco de dados, bem como do tipo de atualização.

> **Note**
>
> A janela da manutenção da instância não se aplica às atualizações do mecanismo.

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados em todas as instâncias de um cluster de banco de dados, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso dos clusters ou do cluster de banco de dados. É possível visualizar ou alterar as configurações da janela de manutenção no Console do Neptune.

Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Melhorias

- O Neptune aguarda mais tempo (dentro do tempo limite especificado da consulta) para que qualquer conflito seja resolvido. Isso reduz o número de exceções de modificações simultâneas que precisam ser tratadas pelo cliente (consulte Erros de consulta).
- Corrigido um problema em que a cardinalidade do Gremlin, às vezes, fazia o mecanismo de aplicação reiniciar.
- Melhorado o desempenho do Gremlin para consultas repetidas `emit.times`.
- Corrigido um problema do Gremlin em que `repeat.until` estava permitindo soluções `.emit` que deveriam ter sido filtradas.
- Melhorado o tratamento de erros no Gremlin.

Atualizações do mecanismo do Amazon Neptune de 19/11/2018

Versão: 1.0.1.0.200264.0

O Amazon Neptune 1.0.1.0.200264.0 está disponível ao público em geral. Todos os novos clusters de banco de dados do Neptune, incluindo aqueles restaurados de snapshots, serão criados no Neptune 1.0.1.0.200264.0 depois que a atualização do mecanismo for concluída para essa região.

Clusters existentes podem ser atualizados para essa versão imediatamente usando as operações de cluster do banco de dados no console ou usando o SDK. É possível usar o seguinte comando da CLI para atualizar um cluster de banco de dados para esta versão imediatamente:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Os clusters de banco de dados do Neptune serão automaticamente atualizados para a versão 1.0.1.0.200264.0 do mecanismo durante as janelas de manutenção do sistema. O cronograma da aplicação das atualizações depende da configuração da região e da janela de manutenção do cluster de banco de dados, bem como do tipo de atualização.

> **Note**
>
> A janela da manutenção da instância não se aplica às atualizações do mecanismo.

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados em todas as instâncias de um cluster de banco de dados, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso dos clusters ou do cluster de banco de dados. É possível visualizar ou alterar as configurações da janela de manutenção no Console do Neptune.

Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Melhorias

- O suporte para the section called “Dicas de consulta” foi adicionado.
- Mensagens de erro aprimoradas para a autenticação do IAM. Para obter mais informações, consulte the section called “Erros de IAM”.
- Desempenho da consulta do SPARQL aprimorada com um alto número de predicados.
- Desempenho do caminho da propriedade SPARQL aperfeiçoado.
- Desempenho do Gremlin aperfeiçoado para mutações condicionais, como o padrão do `fold().coalesce(unfold(), …)`, quando usado com `addV()`, `addE()` e as etapas `property()`.
- Desempenho do Gremlin aperfeiçoado para modulações `by()` e `sack()`.
- Desempenho do Gremlin aperfeiçoado para as etapas `group()` e `groupCount()`.
- Desempenho do Gremlin aperfeiçoado para as etapas `store()`, `sideEffect()` e `cap().unfold()`.
- Suporte aprimorado para restrições de cardinalidade de propriedades únicas do Gremlin.
  - Implementação da cardinalidade única aprimorada para propriedades de borda e propriedades de vértice marcadas como propriedades de cardinalidade únicas.

- Um erro é introduzido se valores de propriedade adicionais forem especificados por uma propriedade de borda existente durante os trabalhos do Neptune Load.

Atualizações do mecanismo do Amazon Neptune de 08/11/2018

Versão: 1.0.1.0.200258.0

O Amazon Neptune 1.0.1.0.200258.0 está disponível ao público em geral. Todos os novos clusters de banco de dados do Neptune, incluindo aqueles restaurados de snapshots, serão criados no Neptune 1.0.1.0.200258.0 depois que a atualização do mecanismo for concluída para essa região.

Clusters existentes podem ser atualizados para essa versão imediatamente usando as operações de cluster do banco de dados no console ou usando o SDK. É possível usar o seguinte comando da CLI para atualizar um cluster de banco de dados para esta versão imediatamente:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Os clusters de banco de dados do Neptune serão automaticamente atualizados para a versão 1.0.1.0.200258.0 do mecanismo durante as janelas de manutenção do sistema. O cronograma da aplicação das atualizações depende da configuração da região e da janela de manutenção do cluster de banco de dados, bem como do tipo de atualização.

> **Note**
>
> A janela da manutenção da instância não se aplica às atualizações do mecanismo.

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados em todas as instâncias de um cluster de banco de dados, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso dos clusters ou do cluster de banco de dados. É possível visualizar ou alterar as configurações da janela de manutenção no Console do Neptune.

Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Melhorias

- O suporte para Dicas de consulta do SPARQL foi adicionado.
- Melhor desempenho para consultas Exists do FILTRO SPARQL (NÃO).
- Melhor desempenho para consultas do SPARQL DESCRIBE.
- Melhor desempenho para repetição até o padrão no Gremlin.
- Melhor desempenho para adicionar bordas no Gremlin.
- Corrigido um problema em que as consultas SPARQL Update DELETE falhavam em alguns casos.
- Corrigido um problema de processamento de tempo limite com o servidor Gremlin WebSocket.

Atualizações do mecanismo do Amazon Neptune de 29/10/2018

Versão: 1,0.1.0.200255.0

O Amazon Neptune 1.0.1.0.200255.0 está disponível ao público em geral. Todos os novos clusters de banco de dados do Neptune, incluindo aqueles restaurados de snapshots, serão criados no Neptune 1.0.1.0.200255.0 depois que a atualização do mecanismo for concluída para essa região.

Clusters existentes podem ser atualizados para essa versão imediatamente usando as operações de cluster do banco de dados no console ou usando o SDK. É possível usar o seguinte comando da CLI para atualizar um cluster de banco de dados para esta versão imediatamente:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Os clusters de banco de dados do Neptune serão automaticamente atualizados para a versão 1.0.1.0.200255.0 do mecanismo durante as janelas de manutenção do sistema. O cronograma da aplicação das atualizações depende da configuração da região e da janela de manutenção do cluster de banco de dados, bem como do tipo de atualização.

> **Note**
>
> A janela da manutenção da instância não se aplica às atualizações do mecanismo.

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados em todas as instâncias de um cluster de banco de dados, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso dos clusters ou do cluster de banco de dados. É possível visualizar ou alterar as configurações da janela de manutenção no Console do Neptune.

Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Melhorias

- Informações de autenticação do IAM adicionadas aos logs de auditoria.
- Suporte adicionado para credenciais temporárias usando as funções do IAM e os perfis de instância.
- Adicionado o encerramento de conexão WebSocket para autenticações do IAM quando a permissão é revogada, ou se o usuário ou função do IAM for excluído.
- Número máximo de conexões WebSocket limitado a 60.000 por instância.
- Melhor desempenho de carregamento em massa para tipos de instância menores.
- Melhor desempenho para consultas que incluem os operadores no Gremlin `and()`, `or()`, `not()`, `drop()`.
- O analisador NTriples agora rejeita URIs inválidas, como URIs que contenham espaço em branco.

Atualizações do mecanismo do Amazon Neptune de 06/09/2018

Versão: 1.0.1.0.200237.0

O Amazon Neptune 1.0.1.0.200237.0 está disponível ao público em geral. Todos os novos clusters de banco de dados do Neptune, incluindo aqueles restaurados de snapshots, serão criados no Neptune 1.0.1.0.200237.0 depois que a atualização do mecanismo for concluída para essa região.

Clusters existentes podem ser atualizados para essa versão imediatamente usando as operações de cluster do banco de dados no console ou usando o SDK. É possível usar o seguinte comando da CLI para atualizar um cluster de banco de dados para esta versão imediatamente:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Os clusters de banco de dados do Neptune serão automaticamente atualizados para a versão 1.0.1.0.200237.0 do mecanismo durante as janelas de manutenção do sistema. O cronograma da aplicação das atualizações depende da configuração da região e da janela de manutenção do cluster de banco de dados, bem como do tipo de atualização.

> **Note**
>
> A janela da manutenção da instância não se aplica às atualizações do mecanismo.

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados em todas as instâncias de um cluster de banco de dados, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso dos clusters ou do cluster de banco de dados. É possível visualizar ou alterar as configurações da janela de manutenção no Console do Neptune.

Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Melhorias

- Corrigido um problema em que ocorriam falhas em algumas consultas `SPARQL COUNT(DISTINCT)`.
- Corrigido um problema em que as consultas `COUNT`, `SUM` e `MIN` com uma cláusula `DISTINCT` esgotavam a memória.
- Corrigido um problema em que os dados do tipo `BLOB` causavam falha em um carregador do Neptune.
- Corrigido um problema em que inserções duplicadas causavam falhas de transação.
- Corrigido um problema em que as consultas `DROP ALL` não podiam ser canceladas.
- Corrigido um problema em que os clientes do Gremlin podiam travar intermitentemente.
- Atualizados todos os códigos de erro para cargas maiores que 150 milhões como `HTTP 400`.
- Melhor desempenho e precisão de consultas `COUNT()` de único padrão de trio.
- Melhor desempenho de consultas `SPARQL UNION` com cláusulas `BIND`.

Atualizações do mecanismo do Amazon Neptune de 24/07/2018

Versão: 1.0.1.0.200236.0

A versão 1.0.1.0.200236.0 do mecanismo do Amazon Neptune está disponível ao público em geral. Todos os novos clusters de banco de dados do Neptune, incluindo aqueles restaurados de snapshots, serão criados no Neptune 1.0.1.0.200236.0 depois que a atualização do mecanismo for concluída para essa região.

Clusters existentes podem ser atualizados para essa versão imediatamente usando as operações de cluster do banco de dados no console ou usando o SDK. É possível usar o seguinte comando da CLI para atualizar um cluster de banco de dados para esta versão imediatamente:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Os clusters de banco de dados do Neptune serão automaticamente atualizados para a versão 1.0.1.0.200236.0 do mecanismo durante as janelas de manutenção do sistema. O cronograma da aplicação das atualizações depende da configuração da região e da janela de manutenção do cluster de banco de dados, bem como do tipo de atualização.

> **Note**
>
> A janela da manutenção da instância não se aplica às atualizações do mecanismo.

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização banco de dados em todas as instâncias de um cluster de banco de dados, ocorrerá um tempo de inatividade de 20 a 30 segundos a alguns minutos. Depois disso, você poderá retomar o uso dos clusters ou do cluster de banco de dados. É possível visualizar ou alterar as configurações da janela de manutenção no Console do Neptune.

Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Melhorias

- Atualização da serialização SPARQL para o tipo de dados `xsd:string`. xsd:string não é mais incluído na serialização JSON, que agora é compatível com outros formatos de saída.

- Correção de manipulação de `xsd:double`/`xsd:float` infinito. Os valores `-INF`, `NaN` e `INF` agora são reconhecidos e tratados adequadamente em todos os formatos do data loader SPARQL, do UPDATE SPARQL 1.1, e do SPARQL 1.1 Query.
- Corrigido um problema em que havia uma falha inesperada em uma consulta do Gremlin com valores de string vazios.
- Corrigido um problema em que havia uma falha inesperada no Gremlin `aggregate()` e `cap()` em um gráfico vazio.
- Corrigido um problema em que as respostas de erros incorretos são retornadas para o Gremlin quando a especificação de cardinalidade é inválida, por exemplo, `.property(set,id,'10')` e `.property(single,id,'10')`.
- Corrigido um problema em que a sintaxe inválida do Gremlin era retornada como uma InternalFailureException.
- Corrigida a ortografia no `TimeLimitExceeededException` para `TimeLimitExceededException`, nas mensagens de erro.
- Alterados os endpoints do SPARQL e do GREMLIN para responder de maneira consistente quando nenhum script for fornecido.
- As mensagens de erro para muitas solicitações simultâneas foram elucidadas.

Atualizações do mecanismo do Amazon Neptune de 22/06/2018

Versão: 1.0.1.0.200233.0

O Amazon Neptune 1.0.1.0.200233.0 está disponível ao público em geral. Todos os novos clusters de banco de dados do Neptune, incluindo aqueles restaurados de snapshots, serão criados no Neptune 1.0.1.0.200233.0 depois que a atualização do mecanismo for concluída para essa região.

Clusters existentes podem ser atualizados para essa versão imediatamente usando as operações de cluster do banco de dados no console ou usando o SDK. É possível usar o seguinte comando da CLI para atualizar um cluster de banco de dados para esta versão imediatamente:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Os clusters de banco de dados do Neptune serão automaticamente atualizados para a versão 1.0.1.0.200233.0 do mecanismo durante as janelas de manutenção do sistema. O cronograma da

aplicação das atualizações depende da configuração da região e da janela de manutenção do cluster de banco de dados, bem como do tipo de atualização.

> **Note**
>
> A janela da manutenção da instância não se aplica às atualizações do mecanismo.

As atualizações são simultaneamente aplicadas a todas as instâncias em um cluster de banco de dados. Como as atualizações exigem a reinicialização do banco de dados em todas as instâncias de um cluster de banco de dados, ocorrerá um tempo de inatividade de vinte a trinta segundos a alguns minutos. Depois disso, você poderá retomar o uso dos clusters ou do cluster de banco de dados. É possível visualizar ou alterar as configurações da janela de manutenção no Console do Neptune.

Em caso de dúvidas ou preocupações, o AWS Support está disponível nos fóruns da comunidade e por meio do AWS Premium Support.

Melhorias

- Correção de um problema em que um grande número de solicitações de carregamento em lote são emitidas em rápida sucessão e resultam em um erro.
- Correção de um problema dependente de dados em que uma consulta pode falhar com um InternalServerError. O exemplo a seguir mostra o tipo de consulta afetada.

```
g.V("my-id123").as("start").outE("knows").has("edgePropertyKey1",
 P.gt(0)).as("myedge").inV()
               .as("end").select("start", "end", "myedge").by("vertexPropertyKey1")
               .by("vertexPropertyKey1").by("edgePropertyKey1")
```

- Correção de um problema em que um cliente Gremlin Java não pode se conectar ao servidor usando a mesma conexão WebSocket após o tempo limite de uma consulta prolongada.
- Correção de um problema em que as sequências recuadas contidas como parte da consulta Gremlin por HTTP ou consultas baseadas em string pela conexão WebSocket não eram processadas corretamente.

Introdução ao uso das APIs do Amazon Neptune

As APIs de gerenciamento do Amazon Neptune são compatíveis com SDKs para criar, gerenciar e excluir instâncias e clusters de banco de dados do Neptune, enquanto as APIs de dados do Neptune são compatíveis com SDKs para carregar dados no grafo, executar consultas, obter informações sobre os dados no grafo e executar operações de machine learning. Essas APIs estão disponíveis em todas as linguagens de SDKs. Ela assina automaticamente as solicitações da API, elas simplificam consideravelmente a integração do Neptune às aplicações.

Esta página fornece informações sobre como usar essas APIs.

Ações do IAM com nomes diferentes dos equivalentes de SDKs das APIs de dados do Neptune

Ao chamar métodos da API do Neptune em um cluster com a autenticação do IAM habilitada, é necessário ter uma política do IAM anexada ao usuário ou ao perfil que está fazendo as chamadas, fornecendo permissões para as ações a serem realizadas. Você define essas permissões na política usando as ações do IAM correspondentes. Também é possível restringir as ações que podem ser realizadas com chaves de condição do IAM.

A maioria das ações do IAM tem o mesmo nome dos métodos de API aos quais elas correspondem, mas alguns métodos na API de dados têm nomes diferentes, pois algumas são compartilhadas por mais de um método. A tabela abaixo lista os métodos de dados e as ações correspondentes do IAM:

Nome da operação da API de dados	Correspondências do IAM	
CancelGremlinQuery (cancel_gremlin_query)	Ação: `neptune-db: CancelQuery`	
CancelLoaderJob (cancel_loader_job)	Ação: `neptune-db:CancelLoaderJob`	
CancelMLDataProcessingJob (cancel_ml_data_processing_job)	Ação: `neptune-db:CancelMLDataProcessingJob`	

Nome da operação da API de dados	Correspondências do IAM	
CancelMLModelTrainingJob (cancel_ml_model_training_job)	Ação: `neptune-db:CancelMLModelTrainingJob`	
CancelOpenCypherQuery (cancel_open_cypher_query)	Ação: `neptune-db:` **`CancelQuery`**	
CreateMLEndpoint (create_ml_endpoint)	Ação: `neptune-db:CreateMLEndpoint`	
DeleteMLEndpoint (delete_ml_endpoint)	Ação: `neptune-db:DeleteMLEndpoint`	
DeletePropertygraphStatistics (delete_propertygraph_statistics)	Ação: `neptune-db:` **`DeleteStatistics`**	
DeleteSparqlStatistics (delete_sparql_statistics)	Ação: `neptune-db:` **`DeleteStatistics`**	
ExecuteFastReset execute_fast_reset()	Ação: `neptune-db:` **`ResetDatabase`**	
ExecuteGremlinExplainQuery (execute_gremlin_explain_query)	Ações: • `neptune-db:` **`ReadDataViaQuery`** • `neptune-db:` **`WriteDataViaQuery`** • `neptune-db:` **`DeleteDataViaQuery`** Chave de condição: `neptune-db:QueryLanguage:Gremlin`	

Nome da operação da API de dados	Correspondências do IAM	
ExecuteGremlinProfileQuery (execute_gremlin_profile_query)	Ação: `neptune-db:` **`ReadDataViaQuery`** Chave de condição: `neptune-db:QueryLanguage:Gremlin`	
ExecuteGremlinQuery (execute_gremlin_query)	Ações: • `neptune-db:` **`ReadDataViaQuery`** • `neptune-db:` **`WriteDataViaQuery`** • `neptune-db:` **`DeleteDataViaQuery`** Chave de condição: `neptune-db:QueryLanguage:Gremlin`	
ExecuteOpenCypherExplainQuery (execute_open_cypher_explain_query)	Ação: `neptune-db:` **`ReadDataViaQuery`** Chave de condição: `neptune-db:QueryLanguage:OpenCypher`	

Nome da operação da API de dados	Correspondências do IAM	
ExecuteOpenCypherQuery (execute_open_cypher_query)	Ações: • `neptune-db:` **`ReadDataViaQuery`** • `neptune-db:` **`WriteDataViaQuery`** • `neptune-db:` **`DeleteDataViaQuery`** Chave de condição: `neptune-db:QueryLanguage:OpenCypher`	
GetEngineStatus (get_engine_status)	Ação: `neptune-db:GetEngineStatus`	
GetGremlinQueryStatus (get_gremlin_query_status)	Ação: `neptune-db:` **`:GetQueryStatus`** Chave de condição: `neptune-db:QueryLanguage:Gremlin`	
GetLoaderJobStatus (get_loader_job_status)	Ação: `neptune-db:GetLoaderJobStatus`	
GetMLDataProcessingJob (get_ml_data_processing_job)	Ação: `neptune-db:` **`GetMLDataProcessingJobStatus`**	
GetMLEndpoint (get_ml_endpoint)	Ação: `neptune-db:` **`GetMLEndpointStatus`**	

Nome da operação da API de dados	Correspondências do IAM	
GetMLModelTrainingJob (get_ml_model_training_job)	Ação: `neptune-db: GetMLModelTrainingJobStatus`	
GetMLModelTransformJob (get_ml_model_transform_job)	Ação: `neptune-db: GetMLModelTransformJobStatus`	
GetOpenCypherQueryStatus (get_open_cypher_query_status)	Ação: `neptune-db: :GetQueryStatus` Chave de condição: `neptune-db:QueryLanguage:OpenCypher`	
GetPropertygraphStatistics (get_propertygraph_statistics)	Ação: `neptune-db: GetStatisticsStatus`	
GetPropertygraphStream (get_propertygraph_stream)	Ação: `neptune-db: GetStreamRecords` Chaves de condição: • `neptune-db:QueryLanguage:Gremlin` • `neptune-db:QueryLanguage:OpenCypher`	
GetPropertygraphSummary (get_propertygraph_summary)	Ação: `neptune-db: GetGraphSummary`	
GetRDFGraphSummary (get_rdf_graph_summary)	Ação: `neptune-db: GetGraphSummary`	

Nome da operação da API de dados	Correspondências do IAM	
GetSparqlStatistics (get_sparql_statistics)	Ação: `neptune-db:` **`GetStatisticsStatus`**	
GetSparqlStream (get_sparql_stream)	Ação: `neptune-db:` **`:GetStreamRecords`** Chave de condição: `neptune-db:QueryLanguage:Sparql`	
ListGremlinQueries (list_gremlin_queries)	Ação: `neptune-db:` **`:GetQueryStatus`** Chave de condição: `neptune-db:QueryLanguage:Gremlin`	
ListMLEndpoints (list_ml_endpoints)	Ação: `neptune-db:ListMLEndpoints`	
ListMLModelTrainingJobs (list_ml_model_training_jobs)	Ação: `neptune-db:ListMLModelTrainingJobs`	
ListMLModelTransformJobs (list_ml_model_transform_jobs)	Ação: `neptune-db:ListMLModelTransformJobs`	
ListOpenCypherQueries (list_open_cypher_queries)	Ação: `neptune-db:` **`:GetQueryStatus`** Chave de condição: `neptune-db:QueryLanguage:OpenCypher`	

Nome da operação da API de dados	Correspondências do IAM	
ManagePropertygraphStatistics (manage_propertygraph_statistics)	Ação: `neptune-db:` **`ManageStatistics`**	
ManageSparqlStatistics (manage_sparql_statistics)	Ação: `neptune-db:` **`ManageStatistics`**	
StartLoaderJob (start_loader_job)	Ação: `neptune-db:StartLoaderJob`	
StartMLModelDataProcessingJob (start_ml_data_processing_job)	Ação: `neptune-db:StartMLModelDataProcessingJob`	
StartMLModelTrainingJob (start_ml_model_training_job)	Ação: `neptune-db:StartMLModelTrainingJob`	
StartMLModelTransformJob (start_ml_model_transform_job)	Ação: `neptune-db:StartMLModelTransformJob`	

Referência da API de gerenciamento do Amazon Neptune

Este capítulo documenta as APIs do Neptune que você pode usar para gerenciar e manter o cluster de banco de dados do Neptune.

O Neptune opera em clusters de servidores de banco de dados conectados em uma topologia de replicação. Assim, gerenciar o Neptune normalmente envolve implantar alterações em vários servidores e garantir que todas as réplicas do Neptune estejam acompanhando o servidor primário.

Como o Neptune dimensiona o armazenamento subjacente de forma transparente à medida que os dados crescem, gerenciar o Neptune exige relativamente pouco gerenciamento de armazenamento em disco. Da mesma forma, como o Neptune realiza backups contínuos automaticamente, um cluster do Neptune não exige planejamento extensivo nem tempo de inatividade para realizar backups.

Sumário

- CreateDBClusterParameterGroup (ação)
- DeleteDBParameterGroup (ação)
- DeleteDBClusterParameterGroup (ação)
- ModifyDBParameterGroup (ação)
- ModifyDBClusterParameterGroup (ação)
- ResetDBParameterGroup (ação)
- ResetDBClusterParameterGroup (ação)
- DescribeDBParameters (ação)
- DescribeDBParameterGroups (ação)
- DescribeDBClusterParameters (ação)
- DescribeDBClusterParameterGroups (ação)
- DescribeEngineDefaultParameters (ação)
- DescribeEngineDefaultClusterParameters (ação)
- Estruturas:
- Parâmetro (estrutura)
- DBParameterGroup (estrutura)
- DBClusterParameterGroup (estrutura)
- DBParameterGroupStatus (estrutura)

- API da sub-rede do Neptune
  - CreateDBSubnetGroup (ação)
  - DeleteDBSubnetGroup (ação)
  - ModifyDBSubnetGroup (ação)
  - DescribeDBSubnetGroups (ação)
  - Estruturas:
  - Sub-rede (estrutura)
  - DBSubnetGroup (estrutura)
- API de snapshots do Neptune
  - CreateDBClusterSnapshot (ação)
  - DeleteDBClusterSnapshot (ação)
  - CopyDBClusterSnapshot (ação)

- DBEngineVersion (estrutura)
- EngineDefaults (estrutura)
- PendingMaintenanceAction (estrutura)
- ResourcePendingMaintenanceActions (estrutura)
- UpgradeTarget (estrutura)
- Tag (estrutura)

- Tipos de dados comuns do Neptune
  - AvailabilityZone (estrutura)
  - DBSecurityGroupMembership (estrutura)
  - DomainMembership (estrutura)
  - DoubleRange (estrutura)
  - Endpoint (estrutura)
  - Filtro (estrutura)
  - Intervalo (estrutura)
  - ServerlessV2ScalingConfiguration (estrutura)
  - ServerlessV2ScalingConfigurationInfo (estrutura)
  - Fuso horário (estrutura)
  - VpcSecurityGroupMembership (estrutura)
- Exceções do Neptune específicas de APIs individuais
  - AuthorizationAlreadyExistsFault (estrutura)
  - AuthorizationNotFoundFault (estrutura)
  - AuthorizationQuotaExceededFault (estrutura)
  - CertificateNotFoundFault (estrutura)
  - DBClusterAlreadyExistsFault (estrutura)
  - DBClusterNotFoundFault (estrutura)
  - DBClusterParameterGroupNotFoundFault (estrutura)
  - DBClusterQuotaExceededFault (estrutura)
  - DBClusterRoleAlreadyExistsFault (estrutura)
  - DBClusterRoleNotFoundFault (estrutura)
  - DBClusterRoleQuotaExceededFault (estrutura)

- DBClusterSnapshotAlreadyExistsFault (estrutura)
- DBClusterSnapshotNotFoundFault (estrutura)
- DBInstanceAlreadyExistsFault (estrutura)
- DBInstanceNotFoundFault (estrutura)
- DBLogFileNotFoundFault (estrutura)
- DBParameterGroupAlreadyExistsFault (estrutura)
- DBParameterGroupNotFoundFault (estrutura)
- DBParameterGroupQuotaExceededFault (estrutura)
- DBSecurityGroupAlreadyExistsFault (estrutura)
- DBSecurityGroupNotFoundFault (estrutura)
- DBSecurityGroupNotSupportedFault (estrutura)
- DBSecurityGroupQuotaExceededFault (estrutura)
- DBSnapshotAlreadyExistsFault (estrutura)
- DBSnapshotNotFoundFault (estrutura)
- DBSubnetGroupAlreadyExistsFault (estrutura)
- DBSubnetGroupDoesNotCoverEnoughAZs (estrutura)
- DBSubnetGroupNotAllowedFault (estrutura)
- DBSubnetGroupNotFoundFault (estrutura)
- DBSubnetGroupQuotaExceededFault (estrutura)
- DBSubnetQuotaExceededFault (estrutura)
- DBUpgradeDependencyFailureFault (estrutura)
- DomainNotFoundFault (estrutura)
- EventSubscriptionQuotaExceededFault (estrutura)
- GlobalClusterAlreadyExistsFault (estrutura)
- GlobalClusterNotFoundFault (estrutura)
- GlobalClusterQuotaExceededFault (estrutura)
- InstanceQuotaExceededFault (estrutura)
- InsufficientDBClusterCapacityFault (estrutura)
- InsufficientDBInstanceCapacityFault (estrutura)
- InsufficientStorageClusterCapacityFault (estrutura)

- InvalidDBClusterEndpointStateFault (estrutura)
- InvalidDBClusterSnapshotStateFault (estrutura)
- InvalidDBClusterStateFault (estrutura)
- InvalidDBInstanceStateFault (estrutura)
- InvalidDBParameterGroupStateFault (estrutura)
- InvalidDBSecurityGroupStateFault (estrutura)
- InvalidDBSnapshotStateFault (estrutura)
- InvalidDBSubnetGroupFault (estrutura)
- InvalidDBSubnetGroupStateFault (estrutura)
- InvalidDBSubnetStateFault (estrutura)
- InvalidEventSubscriptionStateFault (estrutura)
- InvalidGlobalClusterStateFault (estrutura)
- InvalidOptionGroupStateFault (estrutura)
- InvalidRestoreFault (estrutura)
- InvalidSubnet (estrutura)
- InvalidVPCNetworkStateFault (estrutura)
- KMSKeyNotAccessibleFault (estrutura)
- OptionGroupNotFoundFault (estrutura)
- PointInTimeRestoreNotEnabledFault (estrutura)
- ProvisionedIopsNotAvailableInAZFault (estrutura)
- ResourceNotFoundFault (estrutura)
- SNSInvalidTopicFault (estrutura)
- SNSNoAuthorizationFault (estrutura)
- SNSTopicArnNotFoundFault (estrutura)
- SharedSnapshotQuotaExceededFault (estrutura)
- SnapshotQuotaExceededFault (estrutura)
- SourceNotFoundFault (estrutura)
- StorageQuotaExceededFault (estrutura)
- StorageTypeNotSupportedFault (estrutura)
- SubnetAlreadyInUse (estrutura)

- SubscriptionAlreadyExistFault (estrutura)
- SubscriptionCategoryNotFoundFault (estrutura)
- SubscriptionNotFoundFault (estrutura)

API de clusters de banco de dados do Neptune

Ações:

- CreateDBCluster (ação)
- DeleteDBCluster (ação)
- ModifyDBCluster (ação)
- StartDBCluster (ação)
- StopDBCluster (ação)
- AddRoleToDBCluster (ação)
- RemoveRoleFromDBCluster (ação)
- FailoverDBCluster (ação)
- PromoteReadReplicaDBCluster (ação)
- DescribeDBClusters (ação)

Estruturas:

- DBCluster (estrutura)
- DBClusterMember (estrutura)
- DBClusterRole (estrutura)
- CloudwatchLogsExportConfiguration (estrutura)
- PendingCloudwatchLogsExports (estrutura)
- ClusterPendingModifiedValues (estrutura)

CreateDBCluster (ação)

> O nome da CLI da AWS para essa API é: `create-db-cluster`.

Cria um novo cluster de banco de dados do Amazon Neptune.

É possível usar o parâmetro `ReplicationSourceIdentifier` para criar o cluster de banco de dados como uma réplica de leitura de outro cluster de banco de dados ou de uma instância de banco de dados do Amazon Neptune.

Observe que quando é criado um novo cluster usando `CreateDBCluster` diretamente, a proteção contra exclusão é desabilitada por padrão (quando é criado um novo cluster de produção no console, a proteção contra exclusão é habilitada por padrão). Só será possível excluir um cluster de banco de dados se seu campo `DeletionProtection` estiver definido como `false`.

Solicitação

- AvailabilityZones (na CLI: `--availability-zones`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de zonas de disponibilidade do EC2 nas quais as instâncias no cluster de banco de dados podem ser criadas.
- BackupRetentionPeriod (na CLI: `--backup-retention-period`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número de dias durante os quais os backups automatizados são retidos. Você deve especificar o valor mínimo de 1.

  Padrão: 1

  Restrições:
  - Deve ser um valor de 1 a 35
- CopyTagsToSnapshot (na CLI: `--copy-tags-to-snapshot`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se definido como *`true`*, as tags serão copiadas em qualquer snapshot do cluster de banco de dados criado.
- DatabaseName (na CLI: `--database-name`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome para o seu banco de dados, com até 64 caracteres alfanuméricos. Se um nome não for fornecido, o Amazon Neptune não criará um banco de dados no cluster de banco de dados que você está criando.
- DBClusterIdentifier (na CLI: `--db-cluster-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

O identificador de cluster de banco de dados. Este parâmetro é armazenado como uma string com letras minúsculas.

Restrições:

- Deve conter de 1 a 63 caracteres, incluindo letras, números ou hífens.
- O primeiro caractere deve ser uma letra.
- Não podem terminar com um hífen ou conter dois hífens consecutivos.

Exemplo: `my-cluster1`

- DBClusterParameterGroupName (na CLI: `--db-cluster-parameter-group-name`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do grupo de parâmetros de cluster de banco de dados para associar a este cluster de banco de dados. Se esse argumento for omitido, o padrão será usado.

  Restrições:

  - Se for fornecido, deverá corresponder ao nome de um DBClusterParameterGroup existente.
- DBSubnetGroupName (na CLI: `--db-subnet-group-name`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um grupo de sub-redes de banco de dados a ser associado a esse cluster de banco de dados.

  Restrições: deve corresponder ao nome de um DBSubnetGroup existente. Não deve ser padrão.

  Exemplo: `mySubnetgroup`
- DeletionProtection (na CLI: `--deletion-protection`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Um valor que indica se o cluster de banco de dados tem a proteção contra exclusão habilitada. O banco de dados não pode ser excluído quando a proteção contra exclusão está habilitada. Por padrão, a proteção contra exclusão fica habilitada.
- EnableCloudwatchLogsExports (na CLI: `--enable-cloudwatch-logs-exports`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista dos tipos de logs que esse cluster de banco de dados deve exportar para o CloudWatch Logs. Os tipos de log válidos são: `audit` (para publicar logs de auditoria) e `slowquery` (para publicar logs de consulta lenta). Consulte Publicar logs do Neptune no Amazon CloudWatch Logs.

- EnableIAMDatabaseAuthentication (na CLI: `--enable-iam-database-authentication`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se for definido como `true`, habilitará a autenticação do Amazon Identity and Access Management (IAM) para todo o cluster de banco de dados (isso não pode ser definido na instância).

  Padrão: `false`.

- Engine (na CLI: `--engine`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do mecanismo de banco de dados a ser usado para esse cluster de banco de dados.

  Valores Válidos: `neptune`

- EngineVersion (na CLI: `--engine-version`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O número de versão do mecanismo de banco de dados a ser usado para o novo cluster de banco de dados.

  Exemplo: `1.2.1.0`

- GlobalClusterIdentifier(na CLI:`--global-cluster-identifier`): um GlobalClusterIdentifier, do tipo: `string` (uma string codificada em UTF-8), no mínimo 1 e no máximo 255 ?st?s, correspondendo a esta expressão regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  O ID do banco de dados global do Neptune ao qual esse novo cluster de banco de dados deve ser adicionado.

- KmsKeyId (na CLI: `--kms-key-id`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador da chave do Amazon KMS para um cluster de banco de dados criptografado.

  O identificador de chave KMS é o Amazon Resource Name (ARN) da chave de criptografia KMS. Se você estiver criando um cluster de banco de dados com a mesma conta da Amazon que tem a chave de criptografia do KMS usada para criptografar o novo cluster de banco de dados, use o alias da chave do KMS em vez do ARN da chave de criptografia do KMS.

  Se uma chave de criptografia não for especificada em `KmsKeyId`:

- Se `ReplicationSourceIdentifier` identificar uma origem criptografada, o Amazon Neptune usará a chave de criptografia utilizada para criptografar a origem. Caso contrário, o Amazon Neptune usará a chave de criptografia padrão.
- Se o parâmetro `StorageEncrypted` for true e `ReplicationSourceIdentifier` não for especificado, o Amazon Neptune usará a chave de criptografia padrão.

O Amazon KMS cria a chave de criptografia padrão para a conta da Amazon. A conta da Amazon tem uma chave de criptografia padrão diferente para cada região da Amazon.

Se você criar uma réplica de leitura de um cluster de banco de dados criptografado em outra região da Amazon, é necessário definir `KmsKeyId` como um ID de chave do KMS que seja válido na região da Amazon de destino. Essa chave é usada para criptografar a réplica de leitura nessa região da Amazon.

- Port (na CLI: `--port`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número de porta em que as instâncias no cluster de banco de dados aceitam conexões.

  Padrão: `8182`

- PreferredBackupWindow (na CLI: `--preferred-backup-window`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O intervalo de tempo diário durante o qual os backups automatizados serão criados se eles forem habilitados com o parâmetro `BackupRetentionPeriod`.

  O padrão é uma janela de trinta minutos selecionada aleatoriamente em um bloco de tempo de oito horas para cada região da Amazon. Para ver os blocos de tempo disponíveis, consulte Neptune Maintenance Window no Guia do usuário do Amazon Neptune.

  Restrições:

  - Deve estar no formato `hh24:mi-hh24:mi`.
  - Deve estar expresso no Tempo Universal Coordenado (UTC).
  - Não pode entrar em conflito com a janela de manutenção preferencial.
  - Deve ser, pelo menos, 30 minutos.

- PreferredMaintenanceWindow (na CLI: `--preferred-maintenance-window`): uma string, do tipo: `string` (uma string codificada em UTF-8).

O intervalo de tempo semanal durante o qual a manutenção do sistema pode ocorrer, no Tempo Universal Coordenado (UTC).

Formato: `ddd:hh24:mi-ddd:hh24:mi`

O padrão é uma janela de trinta minutos selecionada aleatoriamente em um bloco de tempo de oito horas para cada região da Amazon, ocorrendo em um dia aleatório da semana. Para ver os blocos de tempo disponíveis, consulte Neptune Maintenance Window no Guia do usuário do Amazon Neptune.

Dias válidos: Mon, Tue, Wed, Thu, Fri, Sat, Sun.

Restrições: janela mínima de 30 minutos.

- PreSignedUrl (na CLI: `--pre-signed-url`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Não há suporte para esse parâmetro atualmente.
- ReplicationSourceIdentifier (na CLI: `--replication-source-identifier`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) da instância ou do cluster de banco de dados de origem, se esse cluster de banco de dados for criado como uma réplica de leitura.
- ServerlessV2ScalingConfiguration (na CLI: `--serverless-v2-scaling-configuration`): um objeto ServerlessV2ScalingConfiguration.

  Contém a configuração de escalabilidade de um cluster de banco de dados do Neptune Serverless.

  Para obter mais informações, consulte Using Amazon Neptune Serverless no Guia do usuário do Amazon Neptune.
- StorageEncrypted (na CLI: `--storage-encrypted`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se o cluster de banco de dados é criptografado.
- StorageType (na CLI: `--storage-type`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo de armazenamento do novo cluster de banco de dados.

Valores válidos:

- **`standard`**: (o padrão). Configura o armazenamento econômico do banco de dados para aplicações com uso moderado a pequeno de E/S. Quando definido como `standard`, o tipo de armazenamento não é retornado na resposta.
- **`iopt1`**: habilita o armazenamento otimizado para E/S desenvolvido para atender às necessidades de workloads de grafos de uso intenso de E/S que exigem preços previsíveis com baixa latência de E/S e throughput de E/S consistente.

  O armazenamento otimizado para E/S do Neptune só está disponível a partir da versão 1.3.0.0 do mecanismo.

- Tags (na CLI: `--tags`): uma matriz de objetos Tag.

  As tags a serem atribuídas ao novo cluster de banco de dados.

- VpcSecurityGroupIds (na CLI: `--vpc-security-group-ids`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de grupos de segurança da VPC do EC2 a serem associados a esse cluster de banco de dados.

Resposta

Contém os detalhes de um cluster de banco de dados do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta em the section called "DescribeDBClusters".

- AllocatedStorage: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  `AllocatedStorage` sempre retorna 1, pois o tamanho de armazenamento de cluster de banco de dados do Neptune não é fixo, mas se ajusta automaticamente conforme necessário.

- AssociatedRoles – Uma matriz de objetos DBClusterRole.

  Fornece uma lista dos perfis do Amazon Identity and Access Management (IAM) que estão associadas ao cluster de banco de dados. Os perfis do IAM associados a uma permissão de concessão do cluster de banco de dados para o cluster de banco de dados para acessar outros serviços da Amazon em seu nome.

- AutomaticRestartTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Hora em que o cluster de banco de dados será reiniciado automaticamente.
- AvailabilityZones: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece a lista de zonas de disponibilidade do EC2 nas quais as instâncias no cluster de banco de dados podem ser criadas.
- BacktrackConsumedChangeRecords: um LongOptional, do tipo: `long` (um valor inteiro assinado de 64 bits).

  Não compatível com o Neptune.
- BacktrackWindow: um LongOptional, do tipo: `long` (um valor inteiro assinado de 64 bits).

  Não compatível com o Neptune.
- BackupRetentionPeriod: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica o número de dias durante os quais os DB snapshots automáticos são retidos.
- Capacity: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Não compatível com o Neptune.
- CloneGroupId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Identifica o grupo de clones ao qual o cluster do banco de dados está associado.
- ClusterCreateTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a hora em que o cluster de banco de dados foi criado, no Tempo Universal Coordenado (UTC).
- CopyTagsToSnapshot: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se definido como *`true`*, as tags serão copiadas em qualquer snapshot do cluster de banco de dados criado.
- CrossAccountClone: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

Se definido como `true`, o cluster de banco de dados poderá ser clonado em várias contas.

- DatabaseName: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém o nome do banco de dados inicial desse cluster de banco de dados que foi fornecido no momento da criação, caso tenha sido especificado quando o cluster de banco de dados foi criado. Esse mesmo nome é retornado durante toda a duração do cluster de banco de dados.

- DBClusterArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) do cluster de banco de dados.

- DBClusterIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um identificador de cluster de banco de dados fornecido pelo usuário. Esse identificador é a chave exclusiva que identifica um cluster de banco de dados.

- DBClusterMembers – Uma matriz de objetos DBClusterMember.

  Fornece a lista de instâncias que compõem o cluster de banco de dados.

- DBClusterParameterGroup: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o nome do grupo de parâmetros de cluster de banco de dados para o cluster de banco de dados.

- DbClusterResourceId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador exclusivo e imutável da região da Amazon para o cluster de banco de dados. Esse identificador é encontrado nas entradas de log do Amazon CloudTrail sempre que a chave do Amazon KMS para o cluster de banco de dados é acessada.

- DBSubnetGroup: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica informações sobre o grupo de sub-redes associado ao cluster de banco de dados, incluindo o nome, a descrição e as sub-redes no grupo de sub-redes.

- DeletionProtection: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica se o cluster de banco de dados tem a proteção contra exclusão ativada ou não. O banco de dados não pode ser excluído quando a proteção contra exclusão está habilitada.

- EarliestBacktrackTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

Não compatível com o Neptune.

- EarliestRestorableTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a data mais antiga para a qual um banco de dados pode ser restaurado com restauração point-in-time.

- EnabledCloudwatchLogsExports: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de tipos de logs que esse cluster de banco de dados está configurado para exportar para o CloudWatch Logs. Os tipos de log válidos são: `audit` (para publicar logs de auditoria no CloudWatch) e slowquery (para publicar logs de consulta lenta no CloudWatch). Consulte Publicar logs do Neptune no Amazon CloudWatch Logs.

- Endpoint: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o endpoint de conexão para a instância principal do cluster de banco de dados.

- Engine: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o nome do mecanismo de banco de dados a ser usado para esse cluster de banco de dados.

- EngineVersion: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Indica a versão do mecanismo do banco de dados.

- GlobalClusterIdentifier: um GlobalClusterIdentifier, do tipo: `string` (uma string codificada em UTF-8), no mínimo 1 e no máximo 255 ?st?s, correspondendo a esta expressão regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  Contém um identificador de cluster de banco de dados global fornecido pelo usuário. Esse identificador é a chave exclusiva que identifica um banco de dados global.

- HostedZoneId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o ID que o Amazon Route 53 atribui ao criar uma zona hospedada.

- IAMDatabaseAuthenticationEnabled: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  True se o mapeamento de contas do Amazon Identity and Access Management (IAM) para contas de banco de dados estiver habilitado. Caso contrário, false.

- IOOptimizedNextAllowedModificationTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Na próxima vez que for possível modificar o cluster de banco de dados para usar o tipo de armazenamento `iopt1`.

- KmsKeyId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Se `StorageEncrypted` for true, o identificador da chave Amazon KMS para o cluster de banco de dados criptografado.

- LatestRestorableTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a data mais recente até a qual um banco de dados pode ser restaurado com restauração pontual.

- MultiAZ: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se o cluster de banco de dados tem instâncias em várias zonas de disponibilidade.

- PendingModifiedValues: um objeto ClusterPendingModifiedValues.

  Esse tipo de dados é usado como um elemento de resposta na operação `ModifyDBCluster` e contém alterações que serão aplicadas durante a próxima janela de manutenção.

- PercentProgress: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o andamento da operação como uma porcentagem.

- Port: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica a porta onde o mecanismo de banco de dados está realizando a recepção.

- PreferredBackupWindow: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o intervalo de tempo diário durante o qual os backups automatizados serão criados se eles estiverem habilitados, conforme determinado por `BackupRetentionPeriod`.

- PreferredMaintenanceWindow: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o período semanal durante o qual pode ocorrer a manutenção do sistema, em Tempo Universal Coordenado (UTC).

- ReaderEndpoint: uma string, do tipo: `string` (uma string codificada em UTF-8).

O endpoint do leitor do cluster de banco de dados. O endpoint do leitor para uma carga do cluster de banco de dados equilibra as conexões entre as réplicas de leitura que estão disponíveis em um cluster de banco de dados. À medida que os clientes solicitam novas conexões ao endpoint do leitor, o Neptune distribui as solicitações de conexão entre as réplicas de leitura no cluster de banco de dados. Essa funcionalidade pode ajudar a equilibrar sua carga de trabalho de leitura entre várias réplicas de leitura em seu cluster de banco de dados.

Se ocorrer um failover e a réplica de leitura à qual você estiver conectado for promovida à instância principal, sua conexão será interrompida. Para continuar a enviar sua carga de trabalho de leitura a outras réplicas de leitura no cluster, reconecte-se ao endpoint do leitor.

- ReadReplicaIdentifiers: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um ou mais identificadores das réplicas de leitura associadas a esse cluster de banco de dados.

- ReplicationSourceIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Não compatível com o Neptune.

- ReplicationType: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Não compatível com o Neptune.

- ServerlessV2ScalingConfiguration: um objeto ServerlessV2ScalingConfigurationInfo.

  Mostra a configuração de escalabilidade de um cluster de banco de dados do Neptune Serverless.

  Para obter mais informações, consulte Using Amazon Neptune Serverless no Guia do usuário do Amazon Neptune.

- Status: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o estado atual desse cluster de banco de dados.

- StorageEncrypted: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se o cluster de banco de dados é criptografado.

- StorageType: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo de armazenamento usado pelo cluster de banco de dados.

  Valores válidos:

- **`standard`**: (o padrão). Fornece armazenamento econômico do banco de dados para aplicações com uso moderado a pequeno de E/S.
- **`iopt1`**: habilita o armazenamento otimizado para E/S desenvolvido para atender às necessidades de workloads de grafos de uso intenso de E/S que exigem preços previsíveis com baixa latência de E/S e throughput de E/S consistente.

  O armazenamento otimizado para E/S do Neptune só está disponível a partir da versão 1.3.0.0 do mecanismo.

- VpcSecurityGroups – Uma matriz de objetos VpcSecurityGroupMembership.

  Fornece uma lista de grupos de segurança da VPC à qual o cluster de banco de dados pertence.

Erros

- DBClusterAlreadyExistsFault
- InsufficientStorageClusterCapacityFault
- DBClusterQuotaExceededFault
- StorageQuotaExceededFault
- DBSubnetGroupNotFoundFault
- InvalidVPCNetworkStateFault
- InvalidDBClusterStateFault
- InvalidDBSubnetGroupStateFault
- InvalidSubnet
- InvalidDBInstanceStateFault
- DBClusterParameterGroupNotFoundFault
- KMSKeyNotAccessibleFault
- DBClusterNotFoundFault
- DBInstanceNotFoundFault
- DBSubnetGroupDoesNotCoverEnoughAZs
- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault

DeleteDBCluster (ação)

O nome da CLI da AWS para essa API é: `delete-db-cluster`.

A ação DeleteDBCluster exclui um cluster de banco de dados provisionado anteriormente. Quando você exclui um cluster de banco de dados, todos os backups automatizados para esse cluster de banco de dados são excluídos e não podem ser recuperados. Os snapshots manuais do cluster de banco de dados especificado não são excluídos.

O cluster de banco de dados não poderá ser excluído se a proteção contra exclusão estiver ativada. Para excluí-lo, é necessário primeiro definir o campo `DeletionProtection` como `False`.

Solicitação

- DBClusterIdentifier (na CLI: `--db-cluster-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador do cluster de banco de dados a ser excluído. Este parâmetro não diferencia maiúsculas de minúsculas.

  Restrições:

  - Deve ser um DBClusterIdentifier existente.
- FinalDBSnapshotIdentifier (na CLI: `--final-db-snapshot-identifier`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador do novo snapshot de cluster de banco de dados criado quando `SkipFinalSnapshot` é definido como `false`.

  > **Note**
  >
  > Especificar esse parâmetro e também definir o parâmetro `SkipFinalShapshot` como true resultará em um erro.

  Restrições:

  - Deve ter de 1 a 255 letras, números ou hifens.
  - O primeiro caractere deve ser uma letra
  - Não podem terminar com um hífen ou conter dois hífens consecutivos

- SkipFinalSnapshot (na CLI: `--skip-final-snapshot`): um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Determina se um snapshot de cluster de banco de dados foi criado antes de o cluster de banco de dados ser excluído. Se `true` for especificado, nenhum snapshot de cluster de banco de dados será criado. Se `false` for especificado, um snapshot de cluster de banco de dados será criado antes de o cluster de banco de dados ser excluído.

  > **Note**
  >
  > Você deverá especificar um parâmetro `FinalDBSnapshotIdentifier` se `SkipFinalSnapshot` for `false`.

  Padrão: `false`

Resposta

Contém os detalhes de um cluster de banco de dados do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta em the section called "DescribeDBClusters".

- AllocatedStorage: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  `AllocatedStorage` sempre retorna 1, pois o tamanho de armazenamento de cluster de banco de dados do Neptune não é fixo, mas se ajusta automaticamente conforme necessário.
- AssociatedRoles – Uma matriz de objetos DBClusterRole.

  Fornece uma lista dos perfis do Amazon Identity and Access Management (IAM) que estão associadas ao cluster de banco de dados. Os perfis do IAM associados a uma permissão de concessão do cluster de banco de dados para o cluster de banco de dados para acessar outros serviços da Amazon em seu nome.
- AutomaticRestartTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Hora em que o cluster de banco de dados será reiniciado automaticamente.
- AvailabilityZones: uma string, do tipo: `string` (uma string codificada em UTF-8).

Fornece a lista de zonas de disponibilidade do EC2 nas quais as instâncias no cluster de banco de dados podem ser criadas.

- BacktrackConsumedChangeRecords: um LongOptional, do tipo: `long` (um valor inteiro assinado de 64 bits).

  Não compatível com o Neptune.

- BacktrackWindow: um LongOptional, do tipo: `long` (um valor inteiro assinado de 64 bits).

  Não compatível com o Neptune.

- BackupRetentionPeriod: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica o número de dias durante os quais os DB snapshots automáticos são retidos.

- Capacity: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Não compatível com o Neptune.

- CloneGroupId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Identifica o grupo de clones ao qual o cluster do banco de dados está associado.

- ClusterCreateTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a hora em que o cluster de banco de dados foi criado, no Tempo Universal Coordenado (UTC).

- CopyTagsToSnapshot: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se definido como *`true`*, as tags serão copiadas em qualquer snapshot do cluster de banco de dados criado.

- CrossAccountClone: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se definido como `true`, o cluster de banco de dados poderá ser clonado em várias contas.

- DatabaseName: uma string, do tipo: `string` (uma string codificada em UTF-8).

Contém o nome do banco de dados inicial desse cluster de banco de dados que foi fornecido no momento da criação, caso tenha sido especificado quando o cluster de banco de dados foi criado. Esse mesmo nome é retornado durante toda a duração do cluster de banco de dados.

- DBClusterArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) do cluster de banco de dados.

- DBClusterIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um identificador de cluster de banco de dados fornecido pelo usuário. Esse identificador é a chave exclusiva que identifica um cluster de banco de dados.

- DBClusterMembers – Uma matriz de objetos DBClusterMember.

  Fornece a lista de instâncias que compõem o cluster de banco de dados.

- DBClusterParameterGroup: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o nome do grupo de parâmetros de cluster de banco de dados para o cluster de banco de dados.

- DbClusterResourceId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador exclusivo e imutável da região da Amazon para o cluster de banco de dados. Esse identificador é encontrado nas entradas de log do Amazon CloudTrail sempre que a chave do Amazon KMS para o cluster de banco de dados é acessada.

- DBSubnetGroup: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica informações sobre o grupo de sub-redes associado ao cluster de banco de dados, incluindo o nome, a descrição e as sub-redes no grupo de sub-redes.

- DeletionProtection: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica se o cluster de banco de dados tem a proteção contra exclusão ativada ou não. O banco de dados não pode ser excluído quando a proteção contra exclusão está habilitada.

- EarliestBacktrackTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Não compatível com o Neptune.

- EarliestRestorableTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

Especifica a data mais antiga para a qual um banco de dados pode ser restaurado com restauração point-in-time.

- EnabledCloudwatchLogsExports: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de tipos de logs que esse cluster de banco de dados está configurado para exportar para o CloudWatch Logs. Os tipos de log válidos são: `audit` (para publicar logs de auditoria no CloudWatch) e slowquery (para publicar logs de consulta lenta no CloudWatch). Consulte Publicar logs do Neptune no Amazon CloudWatch Logs.
- Endpoint: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o endpoint de conexão para a instância principal do cluster de banco de dados.
- Engine: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o nome do mecanismo de banco de dados a ser usado para esse cluster de banco de dados.
- EngineVersion: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Indica a versão do mecanismo do banco de dados.
- GlobalClusterIdentifier: um GlobalClusterIdentifier, do tipo: `string` (uma string codificada em UTF-8), no mínimo 1 e no máximo 255 ?st?s, correspondendo a esta expressão regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  Contém um identificador de cluster de banco de dados global fornecido pelo usuário. Esse identificador é a chave exclusiva que identifica um banco de dados global.
- HostedZoneId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o ID que o Amazon Route 53 atribui ao criar uma zona hospedada.
- IAMDatabaseAuthenticationEnabled: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  True se o mapeamento de contas do Amazon Identity and Access Management (IAM) para contas de banco de dados estiver habilitado. Caso contrário, false.
- IOOptimizedNextAllowedModificationTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Na próxima vez que for possível modificar o cluster de banco de dados para usar o tipo de armazenamento `iopt1`.

- KmsKeyId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Se `StorageEncrypted` for true, o identificador da chave Amazon KMS para o cluster de banco de dados criptografado.
- LatestRestorableTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a data mais recente até a qual um banco de dados pode ser restaurado com restauração pontual.
- MultiAZ: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se o cluster de banco de dados tem instâncias em várias zonas de disponibilidade.
- PendingModifiedValues: um objeto ClusterPendingModifiedValues.

  Esse tipo de dados é usado como um elemento de resposta na operação `ModifyDBCluster` e contém alterações que serão aplicadas durante a próxima janela de manutenção.
- PercentProgress: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o andamento da operação como uma porcentagem.
- Port: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica a porta onde o mecanismo de banco de dados está realizando a recepção.
- PreferredBackupWindow: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o intervalo de tempo diário durante o qual os backups automatizados serão criados se eles estiverem habilitados, conforme determinado por `BackupRetentionPeriod`.
- PreferredMaintenanceWindow: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o período semanal durante o qual pode ocorrer a manutenção do sistema, em Tempo Universal Coordenado (UTC).
- ReaderEndpoint: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O endpoint do leitor do cluster de banco de dados. O endpoint do leitor para uma carga do cluster de banco de dados equilibra as conexões entre as réplicas de leitura que estão disponíveis em um cluster de banco de dados. À medida que os clientes solicitam novas conexões ao endpoint do leitor, o Neptune distribui as solicitações de conexão entre as réplicas de leitura no cluster de banco de dados. Essa funcionalidade pode ajudar a equilibrar sua carga de trabalho de leitura entre várias réplicas de leitura em seu cluster de banco de dados.

Se ocorrer um failover e a réplica de leitura à qual você estiver conectado for promovida à instância principal, sua conexão será interrompida. Para continuar a enviar sua carga de trabalho de leitura a outras réplicas de leitura no cluster, reconecte-se ao endpoint do leitor.

- ReadReplicaIdentifiers: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um ou mais identificadores das réplicas de leitura associadas a esse cluster de banco de dados.

- ReplicationSourceIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Não compatível com o Neptune.

- ReplicationType: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Não compatível com o Neptune.

- ServerlessV2ScalingConfiguration: um objeto ServerlessV2ScalingConfigurationInfo.

  Mostra a configuração de escalabilidade de um cluster de banco de dados do Neptune Serverless.

  Para obter mais informações, consulte Using Amazon Neptune Serverless no Guia do usuário do Amazon Neptune.

- Status: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o estado atual desse cluster de banco de dados.

- StorageEncrypted: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se o cluster de banco de dados é criptografado.

- StorageType: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo de armazenamento usado pelo cluster de banco de dados.

  Valores válidos:

  - **`standard`**: (o padrão). Fornece armazenamento econômico do banco de dados para aplicações com uso moderado a pequeno de E/S.
  - **`iopt1`**: habilita o armazenamento otimizado para E/S desenvolvido para atender às necessidades de workloads de grafos de uso intenso de E/S que exigem preços previsíveis com baixa latência de E/S e throughput de E/S consistente.

O armazenamento otimizado para E/S do Neptune só está disponível a partir da versão 1.3.0.0 do mecanismo.

- VpcSecurityGroups – Uma matriz de objetos VpcSecurityGroupMembership.

  Fornece uma lista de grupos de segurança da VPC à qual o cluster de banco de dados pertence.

Erros

- DBClusterNotFoundFault
- InvalidDBClusterStateFault
- DBClusterSnapshotAlreadyExistsFault
- SnapshotQuotaExceededFault
- InvalidDBClusterSnapshotStateFault

ModifyDBCluster (ação)

O nome da CLI da AWS para essa API é: `modify-db-cluster`.

Modifique uma configuração de um cluster de banco de dados. Você pode alterar um ou mais parâmetros de configuração do banco de dados especificando esses parâmetros e os novos valores na solicitação.

Solicitação

- AllowMajorVersionUpgrade (na CLI: `--allow-major-version-upgrade`): um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Um valor que indica se as atualizações de outras versões principais são permitidas.

  Restrições: é necessário definir o sinalizador allow-major-version-upgrade ao fornecer um parâmetro `EngineVersion` que use uma versão principal diferente da versão atual do cluster de banco de dados.
- ApplyImmediately (na CLI: `--apply-immediately`): um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Um valor que especifica se as modificações feitas nesta solicitação e todas as modificações pendentes serão aplicadas de maneira assíncrona assim que possível, independentemente

da configuração `PreferredMaintenanceWindow` do cluster de banco de dados. Caso esse parâmetro seja definido como `false`, as alterações feitas no cluster de banco de dados serão aplicadas durante a próxima janela de manutenção.

O parâmetro `ApplyImmediately` afeta apenas os valores `NewDBClusterIdentifier`. Se você definir o valor do parâmetro `ApplyImmediately` como falso e, depois, alterar para `NewDBClusterIdentifier`, os valores serão aplicados durante a próxima janela de manutenção. Todas as demais alterações serão aplicadas de imediato, independentemente do valor do parâmetro `ApplyImmediately`.

Padrão: `false`

- BackupRetentionPeriod (na CLI: `--backup-retention-period`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número de dias durante os quais os backups automatizados são retidos. Você deve especificar o valor mínimo de 1.

  Padrão: 1

  Restrições:

  - Deve ser um valor de 1 a 35
- CloudwatchLogsExportConfiguration (na CLI: `--cloudwatch-logs-export-configuration`): um objeto CloudwatchLogsExportConfiguration.

  A configuração para os tipos de log a serem habilitados para exportação para o CloudWatch Logs para um determinado cluster de banco de dados. Consulte Usar a CLI para publicar logs de auditoria do Neptune no CloudWatch Logs.
- CopyTagsToSnapshot (na CLI: `--copy-tags-to-snapshot`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se definido como *`true`*, as tags serão copiadas em qualquer snapshot do cluster de banco de dados criado.
- DBClusterIdentifier (na CLI: `--db-cluster-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador do cluster de banco de dados que está sendo modificado. Esse parâmetro não diferencia maiúsculas de minúsculas.

Restrições:

- Deve ser o identificador de um DBCluster existente.

- DBClusterParameterGroupName (na CLI: `--db-cluster-parameter-group-name`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do grupo de parâmetros do cluster de banco de dados a ser usado no cluster de banco de dados.

- DBInstanceParameterGroupName (na CLI: `--db-instance-parameter-group-name`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do grupo de parâmetros de banco de dados a ser aplicado a todas as instâncias do cluster de banco de dados.

  > **Note**
  >
  > Quando você aplica um grupo de parâmetros usando `DBInstanceParameterGroupName`, as alterações de parâmetros são aplicadas imediatamente, não durante a próxima janela de manutenção.

  Padrão: a configuração de nome existente

  Restrições:

  - O grupo de parâmetros de banco de dados deve estar na mesma família de grupos de parâmetros de banco de dados que a versão de cluster de banco de dados de destino.
  - O parâmetro `DBInstanceParameterGroupName` só é válido em combinação com o parâmetro `AllowMajorVersionUpgrade`.

- DeletionProtection (na CLI: `--deletion-protection`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Um valor que indica se o cluster de banco de dados tem a proteção contra exclusão habilitada. O banco de dados não pode ser excluído quando a proteção contra exclusão está habilitada. Por padrão, a proteção contra exclusão fica desabilitada.

- EnableIAMDatabaseAuthentication (na CLI: `--enable-iam-database-authentication`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

Verdadeiro para habilitar o mapeamento de contas do Amazon Identity and Access Management (IAM) para contas de banco de dados. Caso contrário, falso.

Padrão: `false`

- EngineVersion (na CLI: `--engine-version`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O número da versão do mecanismo de banco de dados para o qual você deseja atualizar. Alterar esse parâmetro resulta em uma interrupção. A alteração será aplicada durante a próxima janela de manutenção, a menos que o parâmetro `ApplyImmediately` seja definido como true.

  Para obter uma lista de versões válidas do mecanismo, consulte Versões do mecanismo do Amazon Neptune ou chame the section called “DescribeDBEngineVersions”.
- NewDBClusterIdentifier (na CLI: `--new-db-cluster-identifier`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O novo identificador do cluster de banco de dados durante a renomeação de um cluster de banco de dados. Esse valor é armazenado como uma string em minúsculas.

  Restrições:

  - Deve conter de 1 a 63 caracteres, incluindo letras, números ou hífens
  - O primeiro caractere deve ser uma letra
  - Não podem terminar com um hífen ou conter dois hífens consecutivos

  Exemplo: `my-cluster2`
- Port (na CLI: `--port`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número da porta na qual o cluster do DB aceita conexões.

  Restrições: o valor deve ser `1150-65535`

  Padrão: a mesma porta que a do cluster de banco de dados original.
- PreferredBackupWindow (na CLI: `--preferred-backup-window`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O intervalo de tempo diário durante o qual os backups automatizados serão criados se eles forem habilitados com o parâmetro `BackupRetentionPeriod`.

O padrão é uma janela de trinta minutos selecionada aleatoriamente em um bloco de tempo de oito horas para cada região da Amazon.

Restrições:

- Deve estar no formato `hh24:mi-hh24:mi`.
- Deve estar expresso no Tempo Universal Coordenado (UTC).
- Não pode entrar em conflito com a janela de manutenção preferencial.
- Deve ser, pelo menos, 30 minutos.

- PreferredMaintenanceWindow (na CLI: `--preferred-maintenance-window`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O intervalo de tempo semanal durante o qual a manutenção do sistema pode ocorrer, no Tempo Universal Coordenado (UTC).

  Formato: `ddd:hh24:mi-ddd:hh24:mi`

  O padrão é uma janela de trinta minutos selecionada aleatoriamente em um bloco de tempo de oito horas para cada região da Amazon, ocorrendo em um dia aleatório da semana.

  Dias válidos: Mon, Tue, Wed, Thu, Fri, Sat, Sun.

  Restrições: janela mínima de 30 minutos.

- ServerlessV2ScalingConfiguration (na CLI: `--serverless-v2-scaling-configuration`): um objeto ServerlessV2ScalingConfiguration.

  Contém a configuração de escalabilidade de um cluster de banco de dados do Neptune Serverless.

  Para obter mais informações, consulte Using Amazon Neptune Serverless no Guia do usuário do Amazon Neptune.

- StorageType (na CLI: `--storage-type`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo de armazenamento a ser associado ao cluster de banco de dados.

  Valores válidos:

- **`standard`**: (o padrão). Configura o armazenamento econômico do banco de dados para aplicações com uso moderado a pequeno de E/S.
- **`iopt1`**: habilita o armazenamento otimizado para E/S desenvolvido para atender às necessidades de workloads de grafos de uso intenso de E/S que exigem preços previsíveis com baixa latência de E/S e throughput de E/S consistente.

  O armazenamento otimizado para E/S do Neptune só está disponível a partir da versão 1.3.0.0 do mecanismo.

- VpcSecurityGroupIds (na CLI: `--vpc-security-group-ids`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  A lista de grupos de segurança da VPC a que o cluster de banco de dados pertencerá.

Resposta

Contém os detalhes de um cluster de banco de dados do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta em the section called "DescribeDBClusters".

- AllocatedStorage: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  `AllocatedStorage` sempre retorna 1, pois o tamanho de armazenamento de cluster de banco de dados do Neptune não é fixo, mas se ajusta automaticamente conforme necessário.

- AssociatedRoles – Uma matriz de objetos DBClusterRole.

  Fornece uma lista dos perfis do Amazon Identity and Access Management (IAM) que estão associadas ao cluster de banco de dados. Os perfis do IAM associados a uma permissão de concessão do cluster de banco de dados para o cluster de banco de dados para acessar outros serviços da Amazon em seu nome.

- AutomaticRestartTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Hora em que o cluster de banco de dados será reiniciado automaticamente.

- AvailabilityZones: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece a lista de zonas de disponibilidade do EC2 nas quais as instâncias no cluster de banco de dados podem ser criadas.

- BacktrackConsumedChangeRecords: um LongOptional, do tipo: `long` (um valor inteiro assinado de 64 bits).

  Não compatível com o Neptune.

- BacktrackWindow: um LongOptional, do tipo: `long` (um valor inteiro assinado de 64 bits).

  Não compatível com o Neptune.

- BackupRetentionPeriod: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica o número de dias durante os quais os DB snapshots automáticos são retidos.

- Capacity: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Não compatível com o Neptune.

- CloneGroupId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Identifica o grupo de clones ao qual o cluster do banco de dados está associado.

- ClusterCreateTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a hora em que o cluster de banco de dados foi criado, no Tempo Universal Coordenado (UTC).

- CopyTagsToSnapshot: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se definido como *`true`*, as tags serão copiadas em qualquer snapshot do cluster de banco de dados criado.

- CrossAccountClone: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se definido como `true`, o cluster de banco de dados poderá ser clonado em várias contas.

- DatabaseName: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém o nome do banco de dados inicial desse cluster de banco de dados que foi fornecido no momento da criação, caso tenha sido especificado quando o cluster de banco de dados foi criado. Esse mesmo nome é retornado durante toda a duração do cluster de banco de dados.

- DBClusterArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

O nome de recurso da Amazon (ARN) do cluster de banco de dados.

- DBClusterIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um identificador de cluster de banco de dados fornecido pelo usuário. Esse identificador é a chave exclusiva que identifica um cluster de banco de dados.

- DBClusterMembers – Uma matriz de objetos DBClusterMember.

  Fornece a lista de instâncias que compõem o cluster de banco de dados.

- DBClusterParameterGroup: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o nome do grupo de parâmetros de cluster de banco de dados para o cluster de banco de dados.

- DbClusterResourceId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador exclusivo e imutável da região da Amazon para o cluster de banco de dados. Esse identificador é encontrado nas entradas de log do Amazon CloudTrail sempre que a chave do Amazon KMS para o cluster de banco de dados é acessada.

- DBSubnetGroup: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica informações sobre o grupo de sub-redes associado ao cluster de banco de dados, incluindo o nome, a descrição e as sub-redes no grupo de sub-redes.

- DeletionProtection: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica se o cluster de banco de dados tem a proteção contra exclusão ativada ou não. O banco de dados não pode ser excluído quando a proteção contra exclusão está habilitada.

- EarliestBacktrackTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Não compatível com o Neptune.

- EarliestRestorableTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a data mais antiga para a qual um banco de dados pode ser restaurado com restauração point-in-time.

- EnabledCloudwatchLogsExports: uma string, do tipo: `string` (uma string codificada em UTF-8).

Uma lista de tipos de logs que esse cluster de banco de dados está configurado para exportar para o CloudWatch Logs. Os tipos de log válidos são: `audit` (para publicar logs de auditoria no CloudWatch) e slowquery (para publicar logs de consulta lenta no CloudWatch). Consulte Publicar logs do Neptune no Amazon CloudWatch Logs.

- Endpoint: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o endpoint de conexão para a instância principal do cluster de banco de dados.

- Engine: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o nome do mecanismo de banco de dados a ser usado para esse cluster de banco de dados.

- EngineVersion: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Indica a versão do mecanismo do banco de dados.

- GlobalClusterIdentifier: um GlobalClusterIdentifier, do tipo: `string` (uma string codificada em UTF-8), no mínimo 1 e no máximo 255 ?st?s, correspondendo a esta expressão regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  Contém um identificador de cluster de banco de dados global fornecido pelo usuário. Esse identificador é a chave exclusiva que identifica um banco de dados global.

- HostedZoneId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o ID que o Amazon Route 53 atribui ao criar uma zona hospedada.

- IAMDatabaseAuthenticationEnabled: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  True se o mapeamento de contas do Amazon Identity and Access Management (IAM) para contas de banco de dados estiver habilitado. Caso contrário, false.

- IOOptimizedNextAllowedModificationTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Na próxima vez que for possível modificar o cluster de banco de dados para usar o tipo de armazenamento `iopt1`.

- KmsKeyId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Se `StorageEncrypted` for true, o identificador da chave Amazon KMS para o cluster de banco de dados criptografado.

- LatestRestorableTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a data mais recente até a qual um banco de dados pode ser restaurado com restauração pontual.
- MultiAZ: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se o cluster de banco de dados tem instâncias em várias zonas de disponibilidade.
- PendingModifiedValues: um objeto ClusterPendingModifiedValues.

  Esse tipo de dados é usado como um elemento de resposta na operação `ModifyDBCluster` e contém alterações que serão aplicadas durante a próxima janela de manutenção.
- PercentProgress: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o andamento da operação como uma porcentagem.
- Port: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica a porta onde o mecanismo de banco de dados está realizando a recepção.
- PreferredBackupWindow: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o intervalo de tempo diário durante o qual os backups automatizados serão criados se eles estiverem habilitados, conforme determinado por `BackupRetentionPeriod`.
- PreferredMaintenanceWindow: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o período semanal durante o qual pode ocorrer a manutenção do sistema, em Tempo Universal Coordenado (UTC).
- ReaderEndpoint: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O endpoint do leitor do cluster de banco de dados. O endpoint do leitor para uma carga do cluster de banco de dados equilibra as conexões entre as réplicas de leitura que estão disponíveis em um cluster de banco de dados. À medida que os clientes solicitam novas conexões ao endpoint do leitor, o Neptune distribui as solicitações de conexão entre as réplicas de leitura no cluster de banco de dados. Essa funcionalidade pode ajudar a equilibrar sua carga de trabalho de leitura entre várias réplicas de leitura em seu cluster de banco de dados.

  Se ocorrer um failover e a réplica de leitura à qual você estiver conectado for promovida à instância principal, sua conexão será interrompida. Para continuar a enviar sua carga de trabalho de leitura a outras réplicas de leitura no cluster, reconecte-se ao endpoint do leitor.

- ReadReplicaIdentifiers: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um ou mais identificadores das réplicas de leitura associadas a esse cluster de banco de dados.

- ReplicationSourceIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Não compatível com o Neptune.

- ReplicationType: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Não compatível com o Neptune.

- ServerlessV2ScalingConfiguration: um objeto ServerlessV2ScalingConfigurationInfo.

  Mostra a configuração de escalabilidade de um cluster de banco de dados do Neptune Serverless.

  Para obter mais informações, consulte Using Amazon Neptune Serverless no Guia do usuário do Amazon Neptune.

- Status: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o estado atual desse cluster de banco de dados.

- StorageEncrypted: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se o cluster de banco de dados é criptografado.

- StorageType: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo de armazenamento usado pelo cluster de banco de dados.

  Valores válidos:

  - **`standard`**: (o padrão). Fornece armazenamento econômico do banco de dados para aplicações com uso moderado a pequeno de E/S.
  - **`iopt1`**: habilita o armazenamento otimizado para E/S desenvolvido para atender às necessidades de workloads de grafos de uso intenso de E/S que exigem preços previsíveis com baixa latência de E/S e throughput de E/S consistente.

    O armazenamento otimizado para E/S do Neptune só está disponível a partir da versão 1.3.0.0 do mecanismo.

- VpcSecurityGroups – Uma matriz de objetos VpcSecurityGroupMembership.

  Fornece uma lista de grupos de segurança da VPC à qual o cluster de banco de dados pertence.

Erros

- DBClusterNotFoundFault
- InvalidDBClusterStateFault
- StorageQuotaExceededFault
- DBSubnetGroupNotFoundFault
- InvalidVPCNetworkStateFault
- InvalidDBSubnetGroupStateFault
- InvalidSubnet
- DBClusterParameterGroupNotFoundFault
- InvalidDBSecurityGroupStateFault
- InvalidDBInstanceStateFault
- DBClusterAlreadyExistsFault
- StorageTypeNotSupportedFault

StartDBCluster (ação)

O nome da CLI da AWS para essa API é: `start-db-cluster`.

Inicia um cluster de banco de dados do Amazon Neptune que foi interrompido usando o console da Amazon, o comando stop-db-cluster da CLI da Amazon ou a API StopDBCluster.

Solicitação

- DBClusterIdentifier (na CLI: `--db-cluster-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador do cluster de banco de dados Neptune a ser iniciado. Este parâmetro é armazenado como uma string com letras minúsculas.

Resposta

Contém os detalhes de um cluster de banco de dados do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta em the section called "DescribeDBClusters".

- AllocatedStorage: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  `AllocatedStorage` sempre retorna 1, pois o tamanho de armazenamento de cluster de banco de dados do Neptune não é fixo, mas se ajusta automaticamente conforme necessário.

- AssociatedRoles – Uma matriz de objetos DBClusterRole.

  Fornece uma lista dos perfis do Amazon Identity and Access Management (IAM) que estão associadas ao cluster de banco de dados. Os perfis do IAM associados a uma permissão de concessão do cluster de banco de dados para o cluster de banco de dados para acessar outros serviços da Amazon em seu nome.

- AutomaticRestartTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Hora em que o cluster de banco de dados será reiniciado automaticamente.

- AvailabilityZones: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece a lista de zonas de disponibilidade do EC2 nas quais as instâncias no cluster de banco de dados podem ser criadas.

- BacktrackConsumedChangeRecords: um LongOptional, do tipo: `long` (um valor inteiro assinado de 64 bits).

  Não compatível com o Neptune.

- BacktrackWindow: um LongOptional, do tipo: `long` (um valor inteiro assinado de 64 bits).

  Não compatível com o Neptune.

- BackupRetentionPeriod: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica o número de dias durante os quais os DB snapshots automáticos são retidos.

- Capacity: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Não compatível com o Neptune.

- CloneGroupId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Identifica o grupo de clones ao qual o cluster do banco de dados está associado.

- ClusterCreateTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

Especifica a hora em que o cluster de banco de dados foi criado, no Tempo Universal Coordenado (UTC).

- CopyTagsToSnapshot: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se definido como *`true`*, as tags serão copiadas em qualquer snapshot do cluster de banco de dados criado.

- CrossAccountClone: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se definido como `true`, o cluster de banco de dados poderá ser clonado em várias contas.

- DatabaseName: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém o nome do banco de dados inicial desse cluster de banco de dados que foi fornecido no momento da criação, caso tenha sido especificado quando o cluster de banco de dados foi criado. Esse mesmo nome é retornado durante toda a duração do cluster de banco de dados.

- DBClusterArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) do cluster de banco de dados.

- DBClusterIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um identificador de cluster de banco de dados fornecido pelo usuário. Esse identificador é a chave exclusiva que identifica um cluster de banco de dados.

- DBClusterMembers – Uma matriz de objetos DBClusterMember.

  Fornece a lista de instâncias que compõem o cluster de banco de dados.

- DBClusterParameterGroup: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o nome do grupo de parâmetros de cluster de banco de dados para o cluster de banco de dados.

- DbClusterResourceId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador exclusivo e imutável da região da Amazon para o cluster de banco de dados. Esse identificador é encontrado nas entradas de log do Amazon CloudTrail sempre que a chave do Amazon KMS para o cluster de banco de dados é acessada.

- DBSubnetGroup: uma string, do tipo: `string` (uma string codificada em UTF-8).

Especifica informações sobre o grupo de sub-redes associado ao cluster de banco de dados, incluindo o nome, a descrição e as sub-redes no grupo de sub-redes.

- DeletionProtection: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica se o cluster de banco de dados tem a proteção contra exclusão ativada ou não. O banco de dados não pode ser excluído quando a proteção contra exclusão está habilitada.

- EarliestBacktrackTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Não compatível com o Neptune.

- EarliestRestorableTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a data mais antiga para a qual um banco de dados pode ser restaurado com restauração point-in-time.

- EnabledCloudwatchLogsExports: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de tipos de logs que esse cluster de banco de dados está configurado para exportar para o CloudWatch Logs. Os tipos de log válidos são: `audit` (para publicar logs de auditoria no CloudWatch) e slowquery (para publicar logs de consulta lenta no CloudWatch). Consulte Publicar logs do Neptune no Amazon CloudWatch Logs.

- Endpoint: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o endpoint de conexão para a instância principal do cluster de banco de dados.

- Engine: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o nome do mecanismo de banco de dados a ser usado para esse cluster de banco de dados.

- EngineVersion: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Indica a versão do mecanismo do banco de dados.

- GlobalClusterIdentifier: um GlobalClusterIdentifier, do tipo: `string` (uma string codificada em UTF-8), no mínimo 1 e no máximo 255 ?st?s, correspondendo a esta expressão regular: `[A-Za-z][0-9A-Za-z-:._]*`.

Contém um identificador de cluster de banco de dados global fornecido pelo usuário. Esse identificador é a chave exclusiva que identifica um banco de dados global.

- HostedZoneId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o ID que o Amazon Route 53 atribui ao criar uma zona hospedada.

- IAMDatabaseAuthenticationEnabled: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  True se o mapeamento de contas do Amazon Identity and Access Management (IAM) para contas de banco de dados estiver habilitado. Caso contrário, false.

- IOOptimizedNextAllowedModificationTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Na próxima vez que for possível modificar o cluster de banco de dados para usar o tipo de armazenamento `iopt1`.

- KmsKeyId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Se `StorageEncrypted` for true, o identificador da chave Amazon KMS para o cluster de banco de dados criptografado.

- LatestRestorableTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a data mais recente até a qual um banco de dados pode ser restaurado com restauração pontual.

- MultiAZ: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se o cluster de banco de dados tem instâncias em várias zonas de disponibilidade.

- PendingModifiedValues: um objeto ClusterPendingModifiedValues.

  Esse tipo de dados é usado como um elemento de resposta na operação `ModifyDBCluster` e contém alterações que serão aplicadas durante a próxima janela de manutenção.

- PercentProgress: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o andamento da operação como uma porcentagem.

- Port: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica a porta onde o mecanismo de banco de dados está realizando a recepção.

- PreferredBackupWindow: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o intervalo de tempo diário durante o qual os backups automatizados serão criados se eles estiverem habilitados, conforme determinado por `BackupRetentionPeriod`.

- PreferredMaintenanceWindow: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o período semanal durante o qual pode ocorrer a manutenção do sistema, em Tempo Universal Coordenado (UTC).

- ReaderEndpoint: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O endpoint do leitor do cluster de banco de dados. O endpoint do leitor para uma carga do cluster de banco de dados equilibra as conexões entre as réplicas de leitura que estão disponíveis em um cluster de banco de dados. À medida que os clientes solicitam novas conexões ao endpoint do leitor, o Neptune distribui as solicitações de conexão entre as réplicas de leitura no cluster de banco de dados. Essa funcionalidade pode ajudar a equilibrar sua carga de trabalho de leitura entre várias réplicas de leitura em seu cluster de banco de dados.

  Se ocorrer um failover e a réplica de leitura à qual você estiver conectado for promovida à instância principal, sua conexão será interrompida. Para continuar a enviar sua carga de trabalho de leitura a outras réplicas de leitura no cluster, reconecte-se ao endpoint do leitor.

- ReadReplicaIdentifiers: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um ou mais identificadores das réplicas de leitura associadas a esse cluster de banco de dados.

- ReplicationSourceIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Não compatível com o Neptune.

- ReplicationType: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Não compatível com o Neptune.

- ServerlessV2ScalingConfiguration: um objeto ServerlessV2ScalingConfigurationInfo.

  Mostra a configuração de escalabilidade de um cluster de banco de dados do Neptune Serverless.

  Para obter mais informações, consulte Using Amazon Neptune Serverless no Guia do usuário do Amazon Neptune.

- Status: uma string, do tipo: `string` (uma string codificada em UTF-8).

Especifica o estado atual desse cluster de banco de dados.

- StorageEncrypted: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se o cluster de banco de dados é criptografado.

- StorageType: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo de armazenamento usado pelo cluster de banco de dados.

  Valores válidos:

  - **`standard`**: (o padrão). Fornece armazenamento econômico do banco de dados para aplicações com uso moderado a pequeno de E/S.
  - **`iopt1`**: habilita o armazenamento otimizado para E/S desenvolvido para atender às necessidades de workloads de grafos de uso intenso de E/S que exigem preços previsíveis com baixa latência de E/S e throughput de E/S consistente.

    O armazenamento otimizado para E/S do Neptune só está disponível a partir da versão 1.3.0.0 do mecanismo.

- VpcSecurityGroups – Uma matriz de objetos VpcSecurityGroupMembership.

  Fornece uma lista de grupos de segurança da VPC à qual o cluster de banco de dados pertence.

Erros

- DBClusterNotFoundFault
- InvalidDBClusterStateFault
- InvalidDBInstanceStateFault

StopDBCluster (ação)

O nome da CLI da AWS para essa API é: `stop-db-cluster`.

Interrompe um cluster de banco de dados do Amazon Neptune. Quando um cluster de banco de dados é interrompido, o Neptune retém os metadados do cluster de banco de dados, incluindo seus endpoints e grupos de parâmetros de banco de dados.

O Neptune também retém os logs de transação para que você possa fazer uma restauração point-in-time, se necessário.

Solicitação

- DBClusterIdentifier (na CLI: `--db-cluster-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador do cluster de banco de dados Neptune a ser interrompido. Este parâmetro é armazenado como uma string com letras minúsculas.

Resposta

Contém os detalhes de um cluster de banco de dados do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta em the section called "DescribeDBClusters".

- AllocatedStorage: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  `AllocatedStorage` sempre retorna 1, pois o tamanho de armazenamento de cluster de banco de dados do Neptune não é fixo, mas se ajusta automaticamente conforme necessário.
- AssociatedRoles – Uma matriz de objetos DBClusterRole.

  Fornece uma lista dos perfis do Amazon Identity and Access Management (IAM) que estão associadas ao cluster de banco de dados. Os perfis do IAM associados a uma permissão de concessão do cluster de banco de dados para o cluster de banco de dados para acessar outros serviços da Amazon em seu nome.
- AutomaticRestartTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Hora em que o cluster de banco de dados será reiniciado automaticamente.
- AvailabilityZones: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece a lista de zonas de disponibilidade do EC2 nas quais as instâncias no cluster de banco de dados podem ser criadas.
- BacktrackConsumedChangeRecords: um LongOptional, do tipo: `long` (um valor inteiro assinado de 64 bits).

  Não compatível com o Neptune.
- BacktrackWindow: um LongOptional, do tipo: `long` (um valor inteiro assinado de 64 bits).

Não compatível com o Neptune.

- BackupRetentionPeriod: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica o número de dias durante os quais os DB snapshots automáticos são retidos.

- Capacity: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Não compatível com o Neptune.

- CloneGroupId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Identifica o grupo de clones ao qual o cluster do banco de dados está associado.

- ClusterCreateTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a hora em que o cluster de banco de dados foi criado, no Tempo Universal Coordenado (UTC).

- CopyTagsToSnapshot: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se definido como *`true`*, as tags serão copiadas em qualquer snapshot do cluster de banco de dados criado.

- CrossAccountClone: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se definido como `true`, o cluster de banco de dados poderá ser clonado em várias contas.

- DatabaseName: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém o nome do banco de dados inicial desse cluster de banco de dados que foi fornecido no momento da criação, caso tenha sido especificado quando o cluster de banco de dados foi criado. Esse mesmo nome é retornado durante toda a duração do cluster de banco de dados.

- DBClusterArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) do cluster de banco de dados.

- DBClusterIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um identificador de cluster de banco de dados fornecido pelo usuário. Esse identificador é a chave exclusiva que identifica um cluster de banco de dados.

- DBClusterMembers – Uma matriz de objetos DBClusterMember.

  Fornece a lista de instâncias que compõem o cluster de banco de dados.

- DBClusterParameterGroup: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o nome do grupo de parâmetros de cluster de banco de dados para o cluster de banco de dados.

- DbClusterResourceId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador exclusivo e imutável da região da Amazon para o cluster de banco de dados. Esse identificador é encontrado nas entradas de log do Amazon CloudTrail sempre que a chave do Amazon KMS para o cluster de banco de dados é acessada.

- DBSubnetGroup: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica informações sobre o grupo de sub-redes associado ao cluster de banco de dados, incluindo o nome, a descrição e as sub-redes no grupo de sub-redes.

- DeletionProtection: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica se o cluster de banco de dados tem a proteção contra exclusão ativada ou não. O banco de dados não pode ser excluído quando a proteção contra exclusão está habilitada.

- EarliestBacktrackTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Não compatível com o Neptune.

- EarliestRestorableTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a data mais antiga para a qual um banco de dados pode ser restaurado com restauração point-in-time.

- EnabledCloudwatchLogsExports: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de tipos de logs que esse cluster de banco de dados está configurado para exportar para o CloudWatch Logs. Os tipos de log válidos são: `audit` (para publicar logs de auditoria no CloudWatch) e slowquery (para publicar logs de consulta lenta no CloudWatch). Consulte Publicar logs do Neptune no Amazon CloudWatch Logs.

- Endpoint: uma string, do tipo: `string` (uma string codificada em UTF-8).

Especifica o endpoint de conexão para a instância principal do cluster de banco de dados.

- Engine: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o nome do mecanismo de banco de dados a ser usado para esse cluster de banco de dados.

- EngineVersion: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Indica a versão do mecanismo do banco de dados.

- GlobalClusterIdentifier: um GlobalClusterIdentifier, do tipo: `string` (uma string codificada em UTF-8), no mínimo 1 e no máximo 255 ?st?s, correspondendo a esta expressão regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  Contém um identificador de cluster de banco de dados global fornecido pelo usuário. Esse identificador é a chave exclusiva que identifica um banco de dados global.

- HostedZoneId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o ID que o Amazon Route 53 atribui ao criar uma zona hospedada.

- IAMDatabaseAuthenticationEnabled: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  True se o mapeamento de contas do Amazon Identity and Access Management (IAM) para contas de banco de dados estiver habilitado. Caso contrário, false.

- IOOptimizedNextAllowedModificationTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Na próxima vez que for possível modificar o cluster de banco de dados para usar o tipo de armazenamento `iopt1`.

- KmsKeyId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Se `StorageEncrypted` for true, o identificador da chave Amazon KMS para o cluster de banco de dados criptografado.

- LatestRestorableTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a data mais recente até a qual um banco de dados pode ser restaurado com restauração pontual.

- MultiAZ: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

Especifica se o cluster de banco de dados tem instâncias em várias zonas de disponibilidade.

- PendingModifiedValues: um objeto ClusterPendingModifiedValues.

  Esse tipo de dados é usado como um elemento de resposta na operação `ModifyDBCluster` e contém alterações que serão aplicadas durante a próxima janela de manutenção.

- PercentProgress: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o andamento da operação como uma porcentagem.

- Port: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica a porta onde o mecanismo de banco de dados está realizando a recepção.

- PreferredBackupWindow: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o intervalo de tempo diário durante o qual os backups automatizados serão criados se eles estiverem habilitados, conforme determinado por `BackupRetentionPeriod`.

- PreferredMaintenanceWindow: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o período semanal durante o qual pode ocorrer a manutenção do sistema, em Tempo Universal Coordenado (UTC).

- ReaderEndpoint: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O endpoint do leitor do cluster de banco de dados. O endpoint do leitor para uma carga do cluster de banco de dados equilibra as conexões entre as réplicas de leitura que estão disponíveis em um cluster de banco de dados. À medida que os clientes solicitam novas conexões ao endpoint do leitor, o Neptune distribui as solicitações de conexão entre as réplicas de leitura no cluster de banco de dados. Essa funcionalidade pode ajudar a equilibrar sua carga de trabalho de leitura entre várias réplicas de leitura em seu cluster de banco de dados.

  Se ocorrer um failover e a réplica de leitura à qual você estiver conectado for promovida à instância principal, sua conexão será interrompida. Para continuar a enviar sua carga de trabalho de leitura a outras réplicas de leitura no cluster, reconecte-se ao endpoint do leitor.

- ReadReplicaIdentifiers: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um ou mais identificadores das réplicas de leitura associadas a esse cluster de banco de dados.

- ReplicationSourceIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

Não compatível com o Neptune.

- ReplicationType: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Não compatível com o Neptune.

- ServerlessV2ScalingConfiguration: um objeto ServerlessV2ScalingConfigurationInfo.

  Mostra a configuração de escalabilidade de um cluster de banco de dados do Neptune Serverless.

  Para obter mais informações, consulte Using Amazon Neptune Serverless no Guia do usuário do Amazon Neptune.

- Status: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o estado atual desse cluster de banco de dados.

- StorageEncrypted: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se o cluster de banco de dados é criptografado.

- StorageType: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo de armazenamento usado pelo cluster de banco de dados.

  Valores válidos:

  - **`standard`**: (o padrão). Fornece armazenamento econômico do banco de dados para aplicações com uso moderado a pequeno de E/S.
  - **`iopt1`**: habilita o armazenamento otimizado para E/S desenvolvido para atender às necessidades de workloads de grafos de uso intenso de E/S que exigem preços previsíveis com baixa latência de E/S e throughput de E/S consistente.

    O armazenamento otimizado para E/S do Neptune só está disponível a partir da versão 1.3.0.0 do mecanismo.

- VpcSecurityGroups – Uma matriz de objetos VpcSecurityGroupMembership.

  Fornece uma lista de grupos de segurança da VPC à qual o cluster de banco de dados pertence.

Erros

- DBClusterNotFoundFault
- InvalidDBClusterStateFault

- InvalidDBInstanceStateFault

AddRoleToDBCluster (ação)

O nome da CLI da AWS para essa API é: `add-role-to-db-cluster`.

Associa um perfil do Identity and Access Management (IAM) a um cluster de banco de dados do Neptune.

Solicitação

- DBClusterIdentifier (na CLI: `--db-cluster-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do cluster de banco de dados a ser associado à função do IAM.
- FeatureName (na CLI: `--feature-name`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do atributo do cluster de banco de dados do Neptune ao qual o perfil do IAM deve ser associado. Para obter a lista de nomes de recursos compatíveis, consulte the section called “DBEngineVersion”.
- RoleArn (na CLI: `--role-arn`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) da função do IAM a ser associada ao cluster de banco de dados do Neptune, por exemplo, `arn:aws:iam::123456789012:role/NeptuneAccessRole`.

Resposta

- Nenhum parâmetro de resposta.

Erros

- DBClusterNotFoundFault
- DBClusterRoleAlreadyExistsFault
- InvalidDBClusterStateFault
- DBClusterRoleQuotaExceededFault

RemoveRoleFromDBCluster (ação)

O nome da CLI da AWS para essa API é: `remove-role-from-db-cluster`.

Desassocia uma função do Identity and Access Management (IAM) de um cluster de banco de dados.

Solicitação

- DBClusterIdentifier (na CLI: `--db-cluster-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do cluster de banco de dados a ser desassociado da função do IAM.
- FeatureName (na CLI: `--feature-name`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do atributo do cluster de banco de dados do qual o perfil do IAM deve ser dissociado. Para obter a lista de nomes de recursos compatíveis, consulte the section called “DescribeDBEngineVersions”.
- RoleArn (na CLI: `--role-arn`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) da função do IAM a ser desassociada do cluster de banco de dados, por exemplo, `arn:aws:iam::123456789012:role/NeptuneAccessRole`.

Resposta

- Nenhum parâmetro de resposta.

Erros

- DBClusterNotFoundFault
- DBClusterRoleNotFoundFault
- InvalidDBClusterStateFault

FailoverDBCluster (ação)

O nome da CLI da AWS para essa API é: `failover-db-cluster`.

Força um failover para um cluster de banco de dados.

Um failover para um cluster de banco de dados promove uma das réplicas de leitura (instâncias somente leitura) no cluster de banco de ados para a instância principal (o gravador do cluster).

O Amazon Neptune fará automaticamente o failover para uma réplica de leitura, se existir, quando ocorrer uma falha na instância principal. Você pode forçar um failover quando quiser simular uma falha de uma instância principal para testes. Como cada instância em um cluster de banco de dados tem seu próprio endereço de endpoint, você precisará limpar e restabelecer todas as conexões existentes que usem esses endereços de endpoint quando o failover estiver concluído.

Solicitação

- DBClusterIdentifier (na CLI: `--db-cluster-identifier`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um identificador de cluster de banco de dados para o qual forçar um failover. Esse parâmetro não diferencia maiúsculas de minúsculas.

  Restrições:

  - Deve ser o identificador de um DBCluster existente.
- TargetDBInstanceIdentifier (na CLI: `--target-db-instance-identifier`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome da instância a ser promovida a instância principal.

  Você deve especificar o identificador da instância para uma réplica de leitura no cluster de banco de dados. Por exemplo, `mydbcluster-replica1`.

Resposta

Contém os detalhes de um cluster de banco de dados do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta em the section called “DescribeDBClusters”.

- AllocatedStorage: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  `AllocatedStorage` sempre retorna 1, pois o tamanho de armazenamento de cluster de banco de dados do Neptune não é fixo, mas se ajusta automaticamente conforme necessário.

- AssociatedRoles – Uma matriz de objetos DBClusterRole.

  Fornece uma lista dos perfis do Amazon Identity and Access Management (IAM) que estão associadas ao cluster de banco de dados. Os perfis do IAM associados a uma permissão de concessão do cluster de banco de dados para o cluster de banco de dados para acessar outros serviços da Amazon em seu nome.
- AutomaticRestartTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Hora em que o cluster de banco de dados será reiniciado automaticamente.
- AvailabilityZones: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece a lista de zonas de disponibilidade do EC2 nas quais as instâncias no cluster de banco de dados podem ser criadas.
- BacktrackConsumedChangeRecords: um LongOptional, do tipo: `long` (um valor inteiro assinado de 64 bits).

  Não compatível com o Neptune.
- BacktrackWindow: um LongOptional, do tipo: `long` (um valor inteiro assinado de 64 bits).

  Não compatível com o Neptune.
- BackupRetentionPeriod: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica o número de dias durante os quais os DB snapshots automáticos são retidos.
- Capacity: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Não compatível com o Neptune.
- CloneGroupId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Identifica o grupo de clones ao qual o cluster do banco de dados está associado.
- ClusterCreateTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a hora em que o cluster de banco de dados foi criado, no Tempo Universal Coordenado (UTC).
- CopyTagsToSnapshot: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

Se definido como *true*, as tags serão copiadas em qualquer snapshot do cluster de banco de dados criado.

- CrossAccountClone: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se definido como `true`, o cluster de banco de dados poderá ser clonado em várias contas.

- DatabaseName: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém o nome do banco de dados inicial desse cluster de banco de dados que foi fornecido no momento da criação, caso tenha sido especificado quando o cluster de banco de dados foi criado. Esse mesmo nome é retornado durante toda a duração do cluster de banco de dados.

- DBClusterArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) do cluster de banco de dados.

- DBClusterIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um identificador de cluster de banco de dados fornecido pelo usuário. Esse identificador é a chave exclusiva que identifica um cluster de banco de dados.

- DBClusterMembers – Uma matriz de objetos DBClusterMember.

  Fornece a lista de instâncias que compõem o cluster de banco de dados.

- DBClusterParameterGroup: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o nome do grupo de parâmetros de cluster de banco de dados para o cluster de banco de dados.

- DbClusterResourceId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador exclusivo e imutável da região da Amazon para o cluster de banco de dados. Esse identificador é encontrado nas entradas de log do Amazon CloudTrail sempre que a chave do Amazon KMS para o cluster de banco de dados é acessada.

- DBSubnetGroup: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica informações sobre o grupo de sub-redes associado ao cluster de banco de dados, incluindo o nome, a descrição e as sub-redes no grupo de sub-redes.

- DeletionProtection: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

Indica se o cluster de banco de dados tem a proteção contra exclusão ativada ou não. O banco de dados não pode ser excluído quando a proteção contra exclusão está habilitada.

- EarliestBacktrackTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Não compatível com o Neptune.

- EarliestRestorableTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a data mais antiga para a qual um banco de dados pode ser restaurado com restauração point-in-time.

- EnabledCloudwatchLogsExports: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de tipos de logs que esse cluster de banco de dados está configurado para exportar para o CloudWatch Logs. Os tipos de log válidos são: `audit` (para publicar logs de auditoria no CloudWatch) e slowquery (para publicar logs de consulta lenta no CloudWatch). Consulte Publicar logs do Neptune no Amazon CloudWatch Logs.

- Endpoint: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o endpoint de conexão para a instância principal do cluster de banco de dados.

- Engine: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o nome do mecanismo de banco de dados a ser usado para esse cluster de banco de dados.

- EngineVersion: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Indica a versão do mecanismo do banco de dados.

- GlobalClusterIdentifier: um GlobalClusterIdentifier, do tipo: `string` (uma string codificada em UTF-8), no mínimo 1 e no máximo 255 ?st?s, correspondendo a esta expressão regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  Contém um identificador de cluster de banco de dados global fornecido pelo usuário. Esse identificador é a chave exclusiva que identifica um banco de dados global.

- HostedZoneId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o ID que o Amazon Route 53 atribui ao criar uma zona hospedada.

- IAMDatabaseAuthenticationEnabled: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  True se o mapeamento de contas do Amazon Identity and Access Management (IAM) para contas de banco de dados estiver habilitado. Caso contrário, false.

- IOOptimizedNextAllowedModificationTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Na próxima vez que for possível modificar o cluster de banco de dados para usar o tipo de armazenamento `iopt1`.

- KmsKeyId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Se `StorageEncrypted` for true, o identificador da chave Amazon KMS para o cluster de banco de dados criptografado.

- LatestRestorableTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a data mais recente até a qual um banco de dados pode ser restaurado com restauração pontual.

- MultiAZ: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se o cluster de banco de dados tem instâncias em várias zonas de disponibilidade.

- PendingModifiedValues: um objeto ClusterPendingModifiedValues.

  Esse tipo de dados é usado como um elemento de resposta na operação `ModifyDBCluster` e contém alterações que serão aplicadas durante a próxima janela de manutenção.

- PercentProgress: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o andamento da operação como uma porcentagem.

- Port: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica a porta onde o mecanismo de banco de dados está realizando a recepção.

- PreferredBackupWindow: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o intervalo de tempo diário durante o qual os backups automatizados serão criados se eles estiverem habilitados, conforme determinado por `BackupRetentionPeriod`.

- PreferredMaintenanceWindow: uma string, do tipo: `string` (uma string codificada em UTF-8).

Especifica o período semanal durante o qual pode ocorrer a manutenção do sistema, em Tempo Universal Coordenado (UTC).

- ReaderEndpoint: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O endpoint do leitor do cluster de banco de dados. O endpoint do leitor para uma carga do cluster de banco de dados equilibra as conexões entre as réplicas de leitura que estão disponíveis em um cluster de banco de dados. À medida que os clientes solicitam novas conexões ao endpoint do leitor, o Neptune distribui as solicitações de conexão entre as réplicas de leitura no cluster de banco de dados. Essa funcionalidade pode ajudar a equilibrar sua carga de trabalho de leitura entre várias réplicas de leitura em seu cluster de banco de dados.

  Se ocorrer um failover e a réplica de leitura à qual você estiver conectado for promovida à instância principal, sua conexão será interrompida. Para continuar a enviar sua carga de trabalho de leitura a outras réplicas de leitura no cluster, reconecte-se ao endpoint do leitor.
- ReadReplicaIdentifiers: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um ou mais identificadores das réplicas de leitura associadas a esse cluster de banco de dados.
- ReplicationSourceIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Não compatível com o Neptune.
- ReplicationType: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Não compatível com o Neptune.
- ServerlessV2ScalingConfiguration: um objeto ServerlessV2ScalingConfigurationInfo.

  Mostra a configuração de escalabilidade de um cluster de banco de dados do Neptune Serverless.

  Para obter mais informações, consulte Using Amazon Neptune Serverless no Guia do usuário do Amazon Neptune.
- Status: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o estado atual desse cluster de banco de dados.
- StorageEncrypted: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se o cluster de banco de dados é criptografado.
- StorageType: uma string, do tipo: `string` (uma string codificada em UTF-8).

O tipo de armazenamento usado pelo cluster de banco de dados.

Valores válidos:

- **`standard`**: (o padrão). Fornece armazenamento econômico do banco de dados para aplicações com uso moderado a pequeno de E/S.
- **`iopt1`**: habilita o armazenamento otimizado para E/S desenvolvido para atender às necessidades de workloads de grafos de uso intenso de E/S que exigem preços previsíveis com baixa latência de E/S e throughput de E/S consistente.

  O armazenamento otimizado para E/S do Neptune só está disponível a partir da versão 1.3.0.0 do mecanismo.

- VpcSecurityGroups – Uma matriz de objetos VpcSecurityGroupMembership.

  Fornece uma lista de grupos de segurança da VPC à qual o cluster de banco de dados pertence.

Erros

- DBClusterNotFoundFault
- InvalidDBClusterStateFault
- InvalidDBInstanceStateFault

PromoteReadReplicaDBCluster (ação)

O nome da CLI da AWS para essa API é: `promote-read-replica-db-cluster`.

Sem suporte.

Solicitação

- DBClusterIdentifier (na CLI: `--db-cluster-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Sem suporte.

Resposta

Contém os detalhes de um cluster de banco de dados do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta em the section called "DescribeDBClusters".

- AllocatedStorage: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  `AllocatedStorage` sempre retorna 1, pois o tamanho de armazenamento de cluster de banco de dados do Neptune não é fixo, mas se ajusta automaticamente conforme necessário.
- AssociatedRoles – Uma matriz de objetos DBClusterRole.

  Fornece uma lista dos perfis do Amazon Identity and Access Management (IAM) que estão associadas ao cluster de banco de dados. Os perfis do IAM associados a uma permissão de concessão do cluster de banco de dados para o cluster de banco de dados para acessar outros serviços da Amazon em seu nome.
- AutomaticRestartTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Hora em que o cluster de banco de dados será reiniciado automaticamente.
- AvailabilityZones: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece a lista de zonas de disponibilidade do EC2 nas quais as instâncias no cluster de banco de dados podem ser criadas.
- BacktrackConsumedChangeRecords: um LongOptional, do tipo: `long` (um valor inteiro assinado de 64 bits).

  Não compatível com o Neptune.
- BacktrackWindow: um LongOptional, do tipo: `long` (um valor inteiro assinado de 64 bits).

  Não compatível com o Neptune.
- BackupRetentionPeriod: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica o número de dias durante os quais os DB snapshots automáticos são retidos.
- Capacity: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Não compatível com o Neptune.
- CloneGroupId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Identifica o grupo de clones ao qual o cluster do banco de dados está associado.

- ClusterCreateTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a hora em que o cluster de banco de dados foi criado, no Tempo Universal Coordenado (UTC).
- CopyTagsToSnapshot: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se definido como *`true`*, as tags serão copiadas em qualquer snapshot do cluster de banco de dados criado.
- CrossAccountClone: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se definido como `true`, o cluster de banco de dados poderá ser clonado em várias contas.
- DatabaseName: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém o nome do banco de dados inicial desse cluster de banco de dados que foi fornecido no momento da criação, caso tenha sido especificado quando o cluster de banco de dados foi criado. Esse mesmo nome é retornado durante toda a duração do cluster de banco de dados.
- DBClusterArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) do cluster de banco de dados.
- DBClusterIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um identificador de cluster de banco de dados fornecido pelo usuário. Esse identificador é a chave exclusiva que identifica um cluster de banco de dados.
- DBClusterMembers – Uma matriz de objetos DBClusterMember.

  Fornece a lista de instâncias que compõem o cluster de banco de dados.
- DBClusterParameterGroup: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o nome do grupo de parâmetros de cluster de banco de dados para o cluster de banco de dados.
- DbClusterResourceId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador exclusivo e imutável da região da Amazon para o cluster de banco de dados. Esse identificador é encontrado nas entradas de log do Amazon CloudTrail sempre que a chave do Amazon KMS para o cluster de banco de dados é acessada.

- DBSubnetGroup: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica informações sobre o grupo de sub-redes associado ao cluster de banco de dados, incluindo o nome, a descrição e as sub-redes no grupo de sub-redes.

- DeletionProtection: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica se o cluster de banco de dados tem a proteção contra exclusão ativada ou não. O banco de dados não pode ser excluído quando a proteção contra exclusão está habilitada.

- EarliestBacktrackTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Não compatível com o Neptune.

- EarliestRestorableTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a data mais antiga para a qual um banco de dados pode ser restaurado com restauração point-in-time.

- EnabledCloudwatchLogsExports: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de tipos de logs que esse cluster de banco de dados está configurado para exportar para o CloudWatch Logs. Os tipos de log válidos são: `audit` (para publicar logs de auditoria no CloudWatch) e slowquery (para publicar logs de consulta lenta no CloudWatch). Consulte Publicar logs do Neptune no Amazon CloudWatch Logs.

- Endpoint: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o endpoint de conexão para a instância principal do cluster de banco de dados.

- Engine: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o nome do mecanismo de banco de dados a ser usado para esse cluster de banco de dados.

- EngineVersion: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Indica a versão do mecanismo do banco de dados.

- GlobalClusterIdentifier: um GlobalClusterIdentifier, do tipo: `string` (uma string codificada em UTF-8), no mínimo 1 e no máximo 255 ?st?s, correspondendo a esta expressão regular: `[A-Za-z][0-9A-Za-z-:._]*`.

Contém um identificador de cluster de banco de dados global fornecido pelo usuário. Esse identificador é a chave exclusiva que identifica um banco de dados global.

- HostedZoneId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o ID que o Amazon Route 53 atribui ao criar uma zona hospedada.

- IAMDatabaseAuthenticationEnabled: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  True se o mapeamento de contas do Amazon Identity and Access Management (IAM) para contas de banco de dados estiver habilitado. Caso contrário, false.

- IOOptimizedNextAllowedModificationTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Na próxima vez que for possível modificar o cluster de banco de dados para usar o tipo de armazenamento `iopt1`.

- KmsKeyId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Se `StorageEncrypted` for true, o identificador da chave Amazon KMS para o cluster de banco de dados criptografado.

- LatestRestorableTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a data mais recente até a qual um banco de dados pode ser restaurado com restauração pontual.

- MultiAZ: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se o cluster de banco de dados tem instâncias em várias zonas de disponibilidade.

- PendingModifiedValues: um objeto ClusterPendingModifiedValues.

  Esse tipo de dados é usado como um elemento de resposta na operação `ModifyDBCluster` e contém alterações que serão aplicadas durante a próxima janela de manutenção.

- PercentProgress: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o andamento da operação como uma porcentagem.

- Port: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica a porta onde o mecanismo de banco de dados está realizando a recepção.

- PreferredBackupWindow: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o intervalo de tempo diário durante o qual os backups automatizados serão criados se eles estiverem habilitados, conforme determinado por `BackupRetentionPeriod`.

- PreferredMaintenanceWindow: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o período semanal durante o qual pode ocorrer a manutenção do sistema, em Tempo Universal Coordenado (UTC).

- ReaderEndpoint: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O endpoint do leitor do cluster de banco de dados. O endpoint do leitor para uma carga do cluster de banco de dados equilibra as conexões entre as réplicas de leitura que estão disponíveis em um cluster de banco de dados. À medida que os clientes solicitam novas conexões ao endpoint do leitor, o Neptune distribui as solicitações de conexão entre as réplicas de leitura no cluster de banco de dados. Essa funcionalidade pode ajudar a equilibrar sua carga de trabalho de leitura entre várias réplicas de leitura em seu cluster de banco de dados.

  Se ocorrer um failover e a réplica de leitura à qual você estiver conectado for promovida à instância principal, sua conexão será interrompida. Para continuar a enviar sua carga de trabalho de leitura a outras réplicas de leitura no cluster, reconecte-se ao endpoint do leitor.

- ReadReplicaIdentifiers: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um ou mais identificadores das réplicas de leitura associadas a esse cluster de banco de dados.

- ReplicationSourceIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Não compatível com o Neptune.

- ReplicationType: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Não compatível com o Neptune.

- ServerlessV2ScalingConfiguration: um objeto ServerlessV2ScalingConfigurationInfo.

  Mostra a configuração de escalabilidade de um cluster de banco de dados do Neptune Serverless.

  Para obter mais informações, consulte Using Amazon Neptune Serverless no Guia do usuário do Amazon Neptune.

- Status: uma string, do tipo: `string` (uma string codificada em UTF-8).

Especifica o estado atual desse cluster de banco de dados.

- StorageEncrypted: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se o cluster de banco de dados é criptografado.

- StorageType: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo de armazenamento usado pelo cluster de banco de dados.

  Valores válidos:

  - **`standard`**: (o padrão). Fornece armazenamento econômico do banco de dados para aplicações com uso moderado a pequeno de E/S.
  - **`iopt1`**: habilita o armazenamento otimizado para E/S desenvolvido para atender às necessidades de workloads de grafos de uso intenso de E/S que exigem preços previsíveis com baixa latência de E/S e throughput de E/S consistente.

    O armazenamento otimizado para E/S do Neptune só está disponível a partir da versão 1.3.0.0 do mecanismo.

- VpcSecurityGroups – Uma matriz de objetos VpcSecurityGroupMembership.

  Fornece uma lista de grupos de segurança da VPC à qual o cluster de banco de dados pertence.

Erros

- DBClusterNotFoundFault
- InvalidDBClusterStateFault

DescribeDBClusters (ação)

O nome da CLI da AWS para essa API é: `describe-db-clusters`.

Retorna informações sobre clusters de banco de dados provisionados e oferece suporte à paginação.

> **Note**
>
> Essa operação também pode retornar informações para clusters do Amazon RDS e clusters do Amazon DocDB.

Solicitação

- DBClusterIdentifier (na CLI: `--db-cluster-identifier`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador de cluster de banco de dados fornecido pelo usuário. Se esse parâmetro for especificado, somente as informações do cluster de banco de dados especificado serão retornadas. Este parâmetro não diferencia maiúsculas de minúsculas.

  Restrições:

  - Se fornecido, deverá ser um DBClusterIdentifier existente.
- Filters (na CLI: `--filters`): uma matriz de objetos Filtro.

  Um filtro que especifica um ou mais clusters de banco de dados a serem descritos.

  Filtros suportados:

  - `db-cluster-id` – aceita identificadores de cluster de banco de dados e os Nomes de recursos da Amazon (ARNs) do cluster de banco de dados. A lista de resultados incluirá somente informações sobre os clusters de banco de dados identificado por esses ARNs.
  - `engine` - aceita um nome de mecanismo (como `neptune`) e restringe a lista de resultados aos clusters de banco de dados criados por esse mecanismo.

  Por exemplo, para invocar essa API na CLI da Amazon e filtrar para que somente clusters de banco de dados do Neptune sejam exibidos, é possível usar o seguinte comando:

  Example

  ```
  aws neptune describe-db-clusters \
              --filters  Name=engine,Values=neptune
  ```
- Marker (na CLI: `--marker`): uma string, do tipo: `string` (uma string codificada em UTF-8).

Um token de paginação opcional fornecido por uma solicitação the section called "DescribeDBClusters" anterior. Se esse parâmetro for especificado, a resposta incluirá apenas os registros além do marcador, até o valor especificado por `MaxRecords`.

- MaxRecords (na CLI: `--max-records`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número máximo de registros a serem incluídos na resposta. Se existirem mais registros do que o valor `MaxRecords` especificado, um token de paginação chamado de marcador será incluído na resposta para que os resultados restantes possam ser recuperados.

  Padrão: 100

  Restrições: Mínimo 20, máximo 100.

Resposta

- DBClusters – Uma matriz de objetos DBCluster.

  Contém uma lista de clusters de banco de dados para o usuário.
- Marker: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um token de paginação que pode ser usado em uma solicitação DescribeDBClusters subsequente.

Erros

- DBClusterNotFoundFault

Estruturas:

DBCluster (estrutura)

Contém os detalhes de um cluster de banco de dados do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta em the section called "DescribeDBClusters".

Campos

- AllocatedStorage: é um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  `AllocatedStorage` sempre retorna 1, pois o tamanho de armazenamento de cluster de banco de dados do Neptune não é fixo, mas se ajusta automaticamente conforme necessário.

- AssociatedRoles: é uma matriz de objetos DBClusterRole.

  Fornece uma lista dos perfis do Amazon Identity and Access Management (IAM) que estão associadas ao cluster de banco de dados. Os perfis do IAM associados a uma permissão de concessão do cluster de banco de dados para o cluster de banco de dados para acessar outros serviços da Amazon em seu nome.

- AutomaticRestartTime: é um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Hora em que o cluster de banco de dados será reiniciado automaticamente.

- AvailabilityZones: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece a lista de zonas de disponibilidade do EC2 nas quais as instâncias no cluster de banco de dados podem ser criadas.

- BacktrackConsumedChangeRecords: é um LongOptional, do tipo: `long` (um valor inteiro assinado de 64 bits).

  Não compatível com o Neptune.

- BacktrackWindow: é um LongOptional, do tipo: `long` (um valor inteiro assinado de 64 bits).

  Não compatível com o Neptune.

- BackupRetentionPeriod: é um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica o número de dias durante os quais os DB snapshots automáticos são retidos.

- Capacity: é um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Não compatível com o Neptune.

- CloneGroupId: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Identifica o grupo de clones ao qual o cluster do banco de dados está associado.

- ClusterCreateTime: é um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a hora em que o cluster de banco de dados foi criado, no Tempo Universal Coordenado (UTC).
- CopyTagsToSnapshot: é um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se definido como *`true`*, as tags serão copiadas em qualquer snapshot do cluster de banco de dados criado.
- CrossAccountClone: é um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se definido como `true`, o cluster de banco de dados poderá ser clonado em várias contas.
- DatabaseName: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém o nome do banco de dados inicial desse cluster de banco de dados que foi fornecido no momento da criação, caso tenha sido especificado quando o cluster de banco de dados foi criado. Esse mesmo nome é retornado durante toda a duração do cluster de banco de dados.
- DBClusterArn: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) do cluster de banco de dados.
- DBClusterIdentifier: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um identificador de cluster de banco de dados fornecido pelo usuário. Esse identificador é a chave exclusiva que identifica um cluster de banco de dados.
- DBClusterMembers: é uma matriz de objetos DBClusterMember.

  Fornece a lista de instâncias que compõem o cluster de banco de dados.
- DBClusterParameterGroup: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o nome do grupo de parâmetros de cluster de banco de dados para o cluster de banco de dados.
- DbClusterResourceId: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador exclusivo e imutável da região da Amazon para o cluster de banco de dados. Esse identificador é encontrado nas entradas de log do Amazon CloudTrail sempre que a chave do Amazon KMS para o cluster de banco de dados é acessada.

- DBSubnetGroup: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica informações sobre o grupo de sub-redes associado ao cluster de banco de dados, incluindo o nome, a descrição e as sub-redes no grupo de sub-redes.

- DeletionProtection: é um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica se o cluster de banco de dados tem a proteção contra exclusão ativada ou não. O banco de dados não pode ser excluído quando a proteção contra exclusão está habilitada.

- EarliestBacktrackTime: é um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Não compatível com o Neptune.

- EarliestRestorableTime: é um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a data mais antiga para a qual um banco de dados pode ser restaurado com restauração point-in-time.

- EnabledCloudwatchLogsExports: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de tipos de logs que esse cluster de banco de dados está configurado para exportar para o CloudWatch Logs. Os tipos de log válidos são: `audit` (para publicar logs de auditoria no CloudWatch) e slowquery (para publicar logs de consulta lenta no CloudWatch). Consulte Publicar logs do Neptune no Amazon CloudWatch Logs.

- Endpoint: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o endpoint de conexão para a instância principal do cluster de banco de dados.

- Engine: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o nome do mecanismo de banco de dados a ser usado para esse cluster de banco de dados.

- EngineVersion: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Indica a versão do mecanismo do banco de dados.

- GlobalClusterIdentifier: é um GlobalClusterIdentifier, do tipo: `string` (uma string codificada em UTF-8), no mínimo 1 e no máximo 255 ?st?s, correspondendo a esta expressão regular: `[A-Za-z][0-9A-Za-z-:._]*`.

Contém um identificador de cluster de banco de dados global fornecido pelo usuário. Esse identificador é a chave exclusiva que identifica um banco de dados global.

- HostedZoneId: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o ID que o Amazon Route 53 atribui ao criar uma zona hospedada.

- IAMDatabaseAuthenticationEnabled: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  True se o mapeamento de contas do Amazon Identity and Access Management (IAM) para contas de banco de dados estiver habilitado. Caso contrário, false.

- IOOptimizedNextAllowedModificationTime: é um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Na próxima vez que for possível modificar o cluster de banco de dados para usar o tipo de armazenamento `iopt1`.

- KmsKeyId: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Se `StorageEncrypted` for true, o identificador da chave Amazon KMS para o cluster de banco de dados criptografado.

- LatestRestorableTime: é um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a data mais recente até a qual um banco de dados pode ser restaurado com restauração pontual.

- MultiAZ: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se o cluster de banco de dados tem instâncias em várias zonas de disponibilidade.

- PendingModifiedValues: é um objeto ClusterPendingModifiedValues.

  Esse tipo de dados é usado como um elemento de resposta na operação `ModifyDBCluster` e contém alterações que serão aplicadas durante a próxima janela de manutenção.

- PercentProgress: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o andamento da operação como uma porcentagem.

- Port: é um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica a porta onde o mecanismo de banco de dados está realizando a recepção.

- PreferredBackupWindow: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o intervalo de tempo diário durante o qual os backups automatizados serão criados se eles estiverem habilitados, conforme determinado por `BackupRetentionPeriod`.

- PreferredMaintenanceWindow: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o período semanal durante o qual pode ocorrer a manutenção do sistema, em Tempo Universal Coordenado (UTC).

- ReaderEndpoint: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O endpoint do leitor do cluster de banco de dados. O endpoint do leitor para uma carga do cluster de banco de dados equilibra as conexões entre as réplicas de leitura que estão disponíveis em um cluster de banco de dados. À medida que os clientes solicitam novas conexões ao endpoint do leitor, o Neptune distribui as solicitações de conexão entre as réplicas de leitura no cluster de banco de dados. Essa funcionalidade pode ajudar a equilibrar sua carga de trabalho de leitura entre várias réplicas de leitura em seu cluster de banco de dados.

  Se ocorrer um failover e a réplica de leitura à qual você estiver conectado for promovida à instância principal, sua conexão será interrompida. Para continuar a enviar sua carga de trabalho de leitura a outras réplicas de leitura no cluster, reconecte-se ao endpoint do leitor.

- ReadReplicaIdentifiers: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um ou mais identificadores das réplicas de leitura associadas a esse cluster de banco de dados.

- ReplicationSourceIdentifier: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Não compatível com o Neptune.

- ReplicationType: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Não compatível com o Neptune.

- ServerlessV2ScalingConfiguration: é um objeto ServerlessV2ScalingConfigurationInfo.

  Mostra a configuração de escalabilidade de um cluster de banco de dados do Neptune Serverless.

  Para obter mais informações, consulte Using Amazon Neptune Serverless no Guia do usuário do Amazon Neptune.

- Status: é uma string, do tipo: `string` (uma string codificada em UTF-8).

Especifica o estado atual desse cluster de banco de dados.

- StorageEncrypted: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se o cluster de banco de dados é criptografado.

- StorageType: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo de armazenamento usado pelo cluster de banco de dados.

  Valores válidos:

  - **`standard`**: (o padrão). Fornece armazenamento econômico do banco de dados para aplicações com uso moderado a pequeno de E/S.
  - **`iopt1`**: habilita o armazenamento otimizado para E/S desenvolvido para atender às necessidades de workloads de grafos de uso intenso de E/S que exigem preços previsíveis com baixa latência de E/S e throughput de E/S consistente.

    O armazenamento otimizado para E/S do Neptune só está disponível a partir da versão 1.3.0.0 do mecanismo.

- VpcSecurityGroups: é uma matriz de objetos VpcSecurityGroupMembership.

  Fornece uma lista de grupos de segurança da VPC à qual o cluster de banco de dados pertence.

`DBCluster` é usado como o elemento de resposta para:

- CreateDBCluster
- DeleteDBCluster
- FailoverDBCluster
- ModifyDBCluster
- PromoteReadReplicaDBCluster
- RestoreDBClusterFromSnapshot
- RestoreDBClusterToPointInTime
- StartDBCluster
- StopDBCluster

DBClusterMember (estrutura)

Contém informações sobre uma instância que faz parte de um cluster de banco de dados.

Campos

- DBClusterParameterGroupStatus: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o status do grupo de parâmetros de cluster de banco de dados para esse membro do cluster de banco de dados.
- DBInstanceIdentifier: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o identificador da instância para esse membro do cluster de banco de dados.
- IsClusterWriter: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Valor que será `true` se o membro do cluster for a instância principal do cluster de banco de dados. Caso contrário, será `false`.
- PromotionTier: é um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Um valor que especifica a ordem em que uma réplica de leitura é promovida à instância principal após uma falha da instância principal existente.

DBClusterRole (estrutura)

Descreve um perfil do Amazon Identity and Access Management (IAM) que está associado a um cluster de banco de dados.

Campos

- FeatureName: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do recurso associado ao perfil do Amazon Identity and Access Management (IAM). Para obter a lista de nomes de recursos compatíveis, consulte the section called “DescribeDBEngineVersions”.
- RoleArn: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O Nome de recurso da Amazon (ARN) da função do IAM associada ao cluster de banco de dados.
- Status: é uma string, do tipo: `string` (uma string codificada em UTF-8).

Descreve o estado da associação entre a função do IAM e o cluster de banco de dados. A propriedade Status retorna um dos valores a seguir:

- `ACTIVE`: o ARN do perfil do IAM é associado ao cluster de banco de dados e pode ser usado para acessar outros serviços da Amazon em seu nome.
- `PENDING` – o ARN da função do IAM está sendo associado ao cluster de banco de dados.
- `INVALID`: o ARN do perfil do IAM está associado ao cluster de banco de dados, mas o cluster de banco de dados não consegue assumir o perfil do IAM para acessar outros serviços da Amazon em seu nome.

CloudwatchLogsExportConfiguration (estrutura)

A configuração para os tipos de log a serem habilitados para exportação para o CloudWatch Logs para um determinado cluster ou instância de banco de dados.

As matrizes `DisableLogTypes` e `EnableLogTypes` determinam os logs que serão exportados (ou não) exportado para o CloudWatch Logs.

Os tipos de log válidos são: `audit` (para publicar logs de auditoria) e `slowquery` (para publicar logs de consulta lenta). Consulte Publicar logs do Neptune no Amazon CloudWatch Logs.

Campos

- DisableLogTypes: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  A lista de tipos de logs a serem desabilitados.
- EnableLogTypes: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  A lista de tipos de logs a serem habilitados.

PendingCloudwatchLogsExports (estrutura)

Uma lista dos tipos de log cuja configuração ainda está pendente. Em outras palavras, esses tipos de log estão em processo de ativação ou desativação.

Os tipos de log válidos são: `audit` (para publicar logs de auditoria) e `slowquery` (para publicar logs de consulta lenta). Consulte Publicar logs do Neptune no Amazon CloudWatch Logs.

Campos

- LogTypesToDisable: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Tipos de log que estão em processo de habilitação. Depois de habilitados, esses tipos de log são exportados para o CloudWatch Logs.
- LogTypesToEnable: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Tipos de log que estão em processo de desativação. Depois que são desativados, esses tipos de log não são exportados para o CloudWatch Logs.

ClusterPendingModifiedValues (estrutura)

Esse tipo de dados é usado como um elemento de resposta na operação `ModifyDBCluster` e contém alterações que serão aplicadas durante a próxima janela de manutenção.

Campos

- AllocatedStorage: é um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O tamanho de armazenamento alocado em gibibytes (GiB) para mecanismos de banco de dados. Para o Neptune, `AllocatedStorage` sempre exibe 1, pois o tamanho de armazenamento do cluster de banco de dados do Neptune não é fixo, mas se ajusta automaticamente conforme necessário.
- BackupRetentionPeriod: é um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número de dias durante os quais os snapshots automáticos do banco de dados são retidos.
- DBClusterIdentifier: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O valor DBClusterIdentifier para o cluster de banco de dados.
- EngineVersion: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Versão do mecanismo de banco de dados.
- IAMDatabaseAuthenticationEnabled: é um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Um valor que indica se o mapeamento de contas do AWS Identity and Access Management (IAM) para contas do banco de dados está habilitado.

- Iops: é um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O valor das IOPS (operações de entrada/saída por segundo) provisionadas. Essa configuração é somente para clusters de banco de dados Multi-AZ.

- PendingCloudwatchLogsExports: é um objeto PendingCloudwatchLogsExports.

  Essa estrutura `PendingCloudwatchLogsExports` especifica alterações pendentes em quais logs do CloudWatch estão habilitados e quais estão desabilitados.

- StorageType: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  A alteração pendente no tipo de armazenamento do cluster de banco de dados. Valores válidos:

  - **`standard`**: (o padrão). Configura o armazenamento econômico do banco de dados para aplicações com uso moderado a pequeno de E/S.
  - **`iopt1`**: habilita o armazenamento otimizado para E/S desenvolvido para atender às necessidades de workloads de grafos de uso intenso de E/S que exigem preços previsíveis com baixa latência de E/S e throughput de E/S consistente.

    O armazenamento otimizado para E/S do Neptune só está disponível a partir da versão 1.3.0.0 do mecanismo.

API de banco de dados global do Neptune

Ações:

- CreateGlobalCluster (ação)
- DeleteGlobalCluster (ação)
- ModifyGlobalCluster (ação)
- DescribeGlobalClusters (ação)
- FailoverGlobalCluster (ação)
- RemoveFromGlobalCluster (ação)

Estruturas:

- GlobalCluster (estrutura)
- GlobalClusterMember (estrutura)

CreateGlobalCluster (ação)

O nome da CLI da AWS para essa API é: `create-global-cluster`.

Cria uma distribuição de banco de dados global do Neptune entre várias regiões da Amazon. O banco de dados global contém um único cluster principal com capacidade de leitura/gravação e um cluster secundário somente leitura que recebe dados do cluster principal por meio da replicação de alta velocidade realizada pelo subsistema de armazenamento do Neptune.

É possível criar um banco de dados global inicialmente vazio e, depois, adicionar um cluster principal e um cluster secundário a ele, ou você pode especificar um cluster do Neptune existente durante a operação de criação para se tornar o cluster principal do banco de dados global.

Solicitação

- DatabaseName (na CLI: `--database-name`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do novo banco de dados global (até 64 caracteres alfanuméricos).
- DeletionProtection (na CLI: `--deletion-protection`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  A configuração de proteção contra exclusão para o novo banco de dados global. O banco de dados global não pode ser excluído quando a proteção contra exclusão está habilitada.
- Engine (na CLI: `--engine`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do mecanismo de banco de dados a ser usado no banco de dados global.

  Valores válidos: `neptune`
- EngineVersion (na CLI: `--engine-version`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  A versão do mecanismo do Neptune a ser usada pelo banco de dados global.

  Valores válidos: `1.2.0.0` ou acima.
- GlobalClusterIdentifier (na CLI: `--global-cluster-identifier`): obrigatório: um GlobalClusterIdentifier, do tipo: `string` (uma string codificada em UTF-8), no mínimo 1 e no máximo 255 ?st?s, correspondendo a esta expressão regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  O identificador de cluster do novo cluster de banco de dados global.

- SourceDBClusterIdentifier (na CLI: `--source-db-cluster-identifier`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  (Opcional) O nome do recurso da Amazon (ARN) de um cluster de banco de dados do Neptune existente a ser usado como o cluster principal do novo banco de dados global.
- StorageEncrypted (na CLI: `--storage-encrypted`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  A configuração de criptografia de armazenamento do novo cluster de banco de dados global.

Resposta

Contém os detalhes de um banco de dados global do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta nas ações the section called “CreateGlobalCluster”, the section called “DescribeGlobalClusters”, the section called “ModifyGlobalCluster”, the section called “DeleteGlobalCluster”, the section called “FailoverGlobalCluster” e the section called “RemoveFromGlobalCluster”.

- DeletionProtection: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  A configuração de proteção contra exclusão do banco de dados global.
- Engine: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O mecanismo de banco de dados do Neptune usado pelo banco de dados global (`"neptune"`).
- EngineVersion: uma string, do tipo: `string` (uma string codificada em UTF-8).

  A versão do mecanismo do Neptune usada pelo banco de dados global.
- GlobalClusterArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do recurso da Amazon (ARN) do banco de dados global.
- GlobalClusterIdentifier: um GlobalClusterIdentifier, do tipo: `string` (uma string codificada em UTF-8), no mínimo 1 e no máximo 255 ?st?s, correspondendo a esta expressão regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  Contém um identificador de cluster de banco de dados global fornecido pelo usuário. Esse identificador é a chave exclusiva que identifica um banco de dados global.

- GlobalClusterMembers – Uma matriz de objetos GlobalClusterMember.

  Uma lista de ARNs de cluster e ARNs de instância para todos os clusters de banco de dados que fazem parte do banco de dados global.
- GlobalClusterResourceId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um identificador imutável para o banco de dados global que é exclusivo em todas as regiões. Esse identificador é encontrado nas entradas de log do CloudTrail sempre que a chave do KMS para o cluster de banco de dados é acessada.
- Status: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o estado atual desse banco de dados global.
- StorageEncrypted: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  A configuração de criptografia de armazenamento do banco de dados global.

Erros

- GlobalClusterAlreadyExistsFault
- GlobalClusterQuotaExceededFault
- InvalidDBClusterStateFault
- DBClusterNotFoundFault

DeleteGlobalCluster (ação)

O nome da CLI da AWS para essa API é: `delete-global-cluster`.

Exclui um banco de dados global. O cluster principal e todos os secundários já devem ter sido desanexados ou excluídos primeiro.

Solicitação

- GlobalClusterIdentifier (na CLI: `--global-cluster-identifier`): obrigatório: um GlobalClusterIdentifier, do tipo: `string` (uma string codificada em UTF-8), no mínimo 1 e no máximo 255 ?st?s, correspondendo a esta expressão regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  O identificador do cluster de banco de dados global que está sendo excluído.

Resposta

Contém os detalhes de um banco de dados global do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta nas ações the section called “CreateGlobalCluster”, the section called “DescribeGlobalClusters”, the section called “ModifyGlobalCluster”, the section called “DeleteGlobalCluster”, the section called “FailoverGlobalCluster” e the section called “RemoveFromGlobalCluster”.

- DeletionProtection: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  A configuração de proteção contra exclusão do banco de dados global.
- Engine: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O mecanismo de banco de dados do Neptune usado pelo banco de dados global (`"neptune"`).
- EngineVersion: uma string, do tipo: `string` (uma string codificada em UTF-8).

  A versão do mecanismo do Neptune usada pelo banco de dados global.
- GlobalClusterArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do recurso da Amazon (ARN) do banco de dados global.
- GlobalClusterIdentifier: um GlobalClusterIdentifier, do tipo: `string` (uma string codificada em UTF-8), no mínimo 1 e no máximo 255 ?st?s, correspondendo a esta expressão regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  Contém um identificador de cluster de banco de dados global fornecido pelo usuário. Esse identificador é a chave exclusiva que identifica um banco de dados global.
- GlobalClusterMembers – Uma matriz de objetos GlobalClusterMember.

  Uma lista de ARNs de cluster e ARNs de instância para todos os clusters de banco de dados que fazem parte do banco de dados global.
- GlobalClusterResourceId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um identificador imutável para o banco de dados global que é exclusivo em todas as regiões. Esse identificador é encontrado nas entradas de log do CloudTrail sempre que a chave do KMS para o cluster de banco de dados é acessada.
- Status: uma string, do tipo: `string` (uma string codificada em UTF-8).

Especifica o estado atual desse banco de dados global.

- StorageEncrypted: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  A configuração de criptografia de armazenamento do banco de dados global.

Erros

- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault

ModifyGlobalCluster (ação)

O nome da CLI da AWS para essa API é: `modify-global-cluster`.

Modifique uma configuração de um cluster global do Amazon Neptune. É possível alterar um ou mais parâmetros de configuração do banco de dados especificando esses parâmetros e os novos valores na solicitação.

Solicitação

- AllowMajorVersionUpgrade (na CLI: `--allow-major-version-upgrade`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Um valor que indica se as atualizações de versões principais são permitidas.

  Restrições: será necessário permitir atualizações da versão principal se você especificar um valor para o parâmetro `EngineVersion` que seja uma versão principal diferente da versão atual do cluster de banco de dados.

  Se você atualizar a versão principal de um banco de dados global, os grupos de parâmetros do cluster e da instância de banco de dados serão definidos como os grupos de parâmetros padrão para a nova versão, portanto, você precisará aplicar qualquer grupo de parâmetros personalizado depois de concluir a atualização.
- DeletionProtection (na CLI: `--deletion-protection`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

Indica se o banco de dados global tem a proteção contra exclusão habilitada. O banco de dados global não pode ser excluído quando a proteção contra exclusão está habilitada.

- EngineVersion (na CLI: `--engine-version`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O número da versão do mecanismo de banco de dados para o qual você deseja atualizar. Alterar esse parâmetro ocasiona uma interrupção. A alteração será aplicada durante a próxima janela de manutenção, a menos que `ApplyImmediately` seja habilitado.

  Para listar todas as versões do mecanismo do Neptune disponíveis, use o seguinte comando:

  Example

  ```
  aws neptune describe-db-engine-versions \
            --engine neptune \
            --query '*[]|[?SupportsGlobalDatabases == 'true'].[EngineVersion]'
  ```

- GlobalClusterIdentifier (na CLI: `--global-cluster-identifier`): obrigatório: um GlobalClusterIdentifier, do tipo: `string` (uma string codificada em UTF-8), no mínimo 1 e no máximo 255 ?st?s, correspondendo a esta expressão regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  O identificador do cluster de banco de dados do cluster global que está sendo modificado. Esse parâmetro não diferencia maiúsculas de minúsculas.

  Restrições: deve corresponder ao identificador de um cluster de banco de dados global existente.

- NewGlobalClusterIdentifier (na CLI:`--new-global-cluster-identifier`): um GlobalClusterIdentifier, do tipo: `string` (uma string codificada em UTF-8), no mínimo 1 e no máximo 255 ?st?s, correspondendo a esta expressão regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  Um novo identificador de cluster a ser atribuído ao banco de dados global. Esse valor é armazenado como uma string em minúsculas.

  Restrições:

  - Deve conter de 1 a 63 caracteres, incluindo letras, números ou hífens.
  - O primeiro caractere deve ser uma letra.
  - Não pode terminar com um hífen nem conter dois hifens consecutivos

  Exemplo: `my-cluster2`

Resposta

Contém os detalhes de um banco de dados global do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta nas ações the section called “CreateGlobalCluster”, the section called “DescribeGlobalClusters”, the section called “ModifyGlobalCluster”, the section called “DeleteGlobalCluster”, the section called “FailoverGlobalCluster” e the section called “RemoveFromGlobalCluster”.

- DeletionProtection: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  A configuração de proteção contra exclusão do banco de dados global.
- Engine: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O mecanismo de banco de dados do Neptune usado pelo banco de dados global (`"neptune"`).
- EngineVersion: uma string, do tipo: `string` (uma string codificada em UTF-8).

  A versão do mecanismo do Neptune usada pelo banco de dados global.
- GlobalClusterArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do recurso da Amazon (ARN) do banco de dados global.
- GlobalClusterIdentifier: um GlobalClusterIdentifier, do tipo: `string` (uma string codificada em UTF-8), no mínimo 1 e no máximo 255 ?st?s, correspondendo a esta expressão regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  Contém um identificador de cluster de banco de dados global fornecido pelo usuário. Esse identificador é a chave exclusiva que identifica um banco de dados global.
- GlobalClusterMembers – Uma matriz de objetos GlobalClusterMember.

  Uma lista de ARNs de cluster e ARNs de instância para todos os clusters de banco de dados que fazem parte do banco de dados global.
- GlobalClusterResourceId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um identificador imutável para o banco de dados global que é exclusivo em todas as regiões. Esse identificador é encontrado nas entradas de log do CloudTrail sempre que a chave do KMS para o cluster de banco de dados é acessada.
- Status: uma string, do tipo: `string` (uma string codificada em UTF-8).

Especifica o estado atual desse banco de dados global.

- StorageEncrypted: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  A configuração de criptografia de armazenamento do banco de dados global.

Erros

- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault

DescribeGlobalClusters (ação)

O nome da CLI da AWS para essa API é: `describe-global-clusters`.

Exibe informações sobre clusters de banco de dados global do Neptune. Essa API dá suporte à paginação.

Solicitação

- GlobalClusterIdentifier (na CLI:`--global-cluster-identifier`): um GlobalClusterIdentifier, do tipo: `string` (uma string codificada em UTF-8), no mínimo 1 e no máximo 255 ?st?s, correspondendo a esta expressão regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  O identificador de cluster de banco de dados fornecido pelo usuário. Se esse parâmetro for especificado, somente as informações sobre o cluster de banco de dados especificado serão exibidas. Esse parâmetro não diferencia maiúsculas de minúsculas.

  Restrições: se fornecido, deverá corresponder a um identificador de um cluster de banco de dados existente.
- Marker (na CLI: `--marker`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  (Opcional) Um token de paginação gerado por uma chamada anterior a `DescribeGlobalClusters`. Se esse parâmetro for especificado, a resposta incluirá apenas os registros além do marcador, até o número especificado por `MaxRecords`.
- MaxRecords (na CLI: `--max-records`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

O número máximo de registros a serem incluídos na resposta. Se houver mais registros do que o valor `MaxRecords` especificado, um token marcador de paginação será incluído na resposta que você poderá usar para recuperar os resultados restantes.

Padrão: `100`

Restrições: mínimo 20, máximo 100.

Resposta

- GlobalClusters – Uma matriz de objetos GlobalCluster.

  A lista de clusters e instâncias globais gerada por essa solicitação.

- Marker: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um token de paginação. Se esse parâmetro for gerado na resposta, mais registros estarão disponíveis, que podem ser recuperados por meio de uma ou mais chamadas adicionais para `DescribeGlobalClusters`.

Erros

- GlobalClusterNotFoundFault

FailoverGlobalCluster (ação)

O nome da CLI da AWS para essa API é: `failover-global-cluster`.

Inicia o processo de failover para um banco de dados global do Neptune.

Um failover para um banco de dados global do Neptune promove um dos clusters de banco de dados secundários somente leitura ao cluster de banco de dados principal e rebaixa o cluster de banco de dados principal a um cluster de banco de dados secundário (somente leitura). Em outras palavras, a função do cluster de banco de dados principal atual e do cluster de banco de dados secundário de destino selecionado são trocadas. O cluster de banco de dados secundário selecionado pressupõe recursos completos de leitura/gravação para o banco de dados global do Neptune.

> **Note**
>
> Essa ação se aplica somente aos bancos de dados globais do Neptune. Essa ação se destina somente ao uso em bancos de dados globais do Neptune íntegros com clusters de banco de dados do Neptune íntegros e sem interrupções em toda a região, para testar cenários de recuperação de desastres ou reconfigurar a topologia do banco de dados global.

Solicitação

- GlobalClusterIdentifier (na CLI: `--global-cluster-identifier`): obrigatório: um GlobalClusterIdentifier, do tipo: `string` (uma string codificada em UTF-8), no mínimo 1 e no máximo 255 ?st?s, correspondendo a esta expressão regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  Identificador do banco de dados global do Neptune que deve ser submetido a failover. O identificador é a chave exclusiva atribuída pelo usuário quando o banco de dados global do Neptune foi criado. Em outras palavras, é o nome do banco de dados global do qual você deseja fazer failover.

  Restrições: deve corresponder ao identificador de um banco de dados global do Neptune existente.
- TargetDbClusterIdentifier (na CLI: `--target-db-cluster-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do recurso da Amazon (ARN) do cluster de banco de dados do Neptune que você deseja promover ao principal para o banco de dados global.

Resposta

Contém os detalhes de um banco de dados global do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta nas ações the section called “CreateGlobalCluster”, the section called “DescribeGlobalClusters”, the section called “ModifyGlobalCluster”, the section called “DeleteGlobalCluster”, the section called “FailoverGlobalCluster” e the section called “RemoveFromGlobalCluster”.

- DeletionProtection: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

A configuração de proteção contra exclusão do banco de dados global.

- Engine: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O mecanismo de banco de dados do Neptune usado pelo banco de dados global (`"neptune"`).
- EngineVersion: uma string, do tipo: `string` (uma string codificada em UTF-8).

  A versão do mecanismo do Neptune usada pelo banco de dados global.
- GlobalClusterArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do recurso da Amazon (ARN) do banco de dados global.
- GlobalClusterIdentifier: um GlobalClusterIdentifier, do tipo: `string` (uma string codificada em UTF-8), no mínimo 1 e no máximo 255 ?st?s, correspondendo a esta expressão regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  Contém um identificador de cluster de banco de dados global fornecido pelo usuário. Esse identificador é a chave exclusiva que identifica um banco de dados global.
- GlobalClusterMembers – Uma matriz de objetos GlobalClusterMember.

  Uma lista de ARNs de cluster e ARNs de instância para todos os clusters de banco de dados que fazem parte do banco de dados global.
- GlobalClusterResourceId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um identificador imutável para o banco de dados global que é exclusivo em todas as regiões. Esse identificador é encontrado nas entradas de log do CloudTrail sempre que a chave do KMS para o cluster de banco de dados é acessada.
- Status: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o estado atual desse banco de dados global.
- StorageEncrypted: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  A configuração de criptografia de armazenamento do banco de dados global.

Erros

- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault

- InvalidDBClusterStateFault
- DBClusterNotFoundFault

RemoveFromGlobalCluster (ação)

O nome da CLI da AWS para essa API é: `remove-from-global-cluster`.

Separa um cluster de banco de dados do Neptune de um banco de dados global do Neptune. Um cluster secundário se torna um cluster autônomo normal com capacidade de leitura e gravação em vez de ser somente leitura, e não recebe mais dados do cluster principal.

Solicitação

- DbClusterIdentifier (na CLI: `--db-cluster-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do recurso da Amazon (ARN) que identifica o cluster a ser separado do cluster de banco de dados global do Neptune.
- GlobalClusterIdentifier (na CLI: `--global-cluster-identifier`): obrigatório: um GlobalClusterIdentifier, do tipo: `string` (uma string codificada em UTF-8), no mínimo 1 e no máximo 255 ?st?s, correspondendo a esta expressão regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  O identificador do banco de dados global do Neptune do qual separar o cluster de banco de dados do Neptune especificado.

Resposta

Contém os detalhes de um banco de dados global do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta nas ações the section called “CreateGlobalCluster”, the section called “DescribeGlobalClusters”, the section called “ModifyGlobalCluster”, the section called “DeleteGlobalCluster”, the section called “FailoverGlobalCluster” e the section called “RemoveFromGlobalCluster”.

- DeletionProtection: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  A configuração de proteção contra exclusão do banco de dados global.

- Engine: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O mecanismo de banco de dados do Neptune usado pelo banco de dados global (`"neptune"`).
- EngineVersion: uma string, do tipo: `string` (uma string codificada em UTF-8).

  A versão do mecanismo do Neptune usada pelo banco de dados global.
- GlobalClusterArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do recurso da Amazon (ARN) do banco de dados global.
- GlobalClusterIdentifier: um GlobalClusterIdentifier, do tipo: `string` (uma string codificada em UTF-8), no mínimo 1 e no máximo 255 ?st?s, correspondendo a esta expressão regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  Contém um identificador de cluster de banco de dados global fornecido pelo usuário. Esse identificador é a chave exclusiva que identifica um banco de dados global.
- GlobalClusterMembers – Uma matriz de objetos GlobalClusterMember.

  Uma lista de ARNs de cluster e ARNs de instância para todos os clusters de banco de dados que fazem parte do banco de dados global.
- GlobalClusterResourceId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um identificador imutável para o banco de dados global que é exclusivo em todas as regiões. Esse identificador é encontrado nas entradas de log do CloudTrail sempre que a chave do KMS para o cluster de banco de dados é acessada.
- Status: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o estado atual desse banco de dados global.
- StorageEncrypted: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  A configuração de criptografia de armazenamento do banco de dados global.

Erros

- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault
- DBClusterNotFoundFault

Estruturas:

GlobalCluster (estrutura)

Contém os detalhes de um banco de dados global do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta nas ações the section called “CreateGlobalCluster”, the section called “DescribeGlobalClusters”, the section called “ModifyGlobalCluster”, the section called “DeleteGlobalCluster”, the section called “FailoverGlobalCluster” e the section called “RemoveFromGlobalCluster”.

Campos

- DeletionProtection: é um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  A configuração de proteção contra exclusão do banco de dados global.
- Engine: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O mecanismo de banco de dados do Neptune usado pelo banco de dados global (`"neptune"`).
- EngineVersion: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  A versão do mecanismo do Neptune usada pelo banco de dados global.
- GlobalClusterArn: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do recurso da Amazon (ARN) do banco de dados global.
- GlobalClusterIdentifier: é um GlobalClusterIdentifier, do tipo: `string` (uma string codificada em UTF-8), no mínimo 1 e no máximo 255 ?st?s, correspondendo a esta expressão regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  Contém um identificador de cluster de banco de dados global fornecido pelo usuário. Esse identificador é a chave exclusiva que identifica um banco de dados global.
- GlobalClusterMembers: é uma matriz de objetos GlobalClusterMember.

  Uma lista de ARNs de cluster e ARNs de instância para todos os clusters de banco de dados que fazem parte do banco de dados global.
- GlobalClusterResourceId: é uma string, do tipo: `string` (uma string codificada em UTF-8).

Um identificador imutável para o banco de dados global que é exclusivo em todas as regiões. Esse identificador é encontrado nas entradas de log do CloudTrail sempre que a chave do KMS para o cluster de banco de dados é acessada.

- Status: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o estado atual desse banco de dados global.

- StorageEncrypted: é um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  A configuração de criptografia de armazenamento do banco de dados global.

`GlobalCluster` é usado como o elemento de resposta para:

- CreateGlobalCluster
- ModifyGlobalCluster
- DeleteGlobalCluster
- RemoveFromGlobalCluster
- FailoverGlobalCluster

GlobalClusterMember (estrutura)

Uma estrutura de dados com informações sobre quaisquer clusters principais e secundários associados a um banco de dados global do Neptune.

Campos

- DBClusterArn: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do recurso da Amazon (ARN) de cada cluster do Neptune.

- IsWriter: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se o cluster do Neptune é o cluster principal (ou seja, tem capacidade de leitura e gravação) do banco de dados global do Neptune ao qual está associado.

- Readers: é uma string, do tipo: `string` (uma string codificada em UTF-8).

O nome do recurso da Amazon (ARN) de cada cluster secundário somente leitura associado ao banco de dados global do Neptune.

API de instâncias do Neptune

Ações:

- CreateDBInstance (ação)
- DeleteDBInstance (ação)
- ModifyDBInstance (ação)
- RebootDBInstance (ação)
- DescribeDBInstances (ação)
- DescribeOrderableDBInstanceOptions (ação)
- DescribeValidDBInstanceModifications (ação)

Estruturas:

- DBInstance (estrutura)
- DBInstanceStatusInfo (estrutura)
- OrderableDBInstanceOption (estrutura)
- PendingModifiedValues (estrutura)
- ValidStorageOptions (estrutura)
- ValidDBInstanceModificationsMessage (estrutura)

CreateDBInstance (ação)

O nome da CLI da AWS para essa API é: `create-db-instance`.

Cria uma nova instância de banco de dados.

Solicitação

- AutoMinorVersionUpgrade (na CLI: `--auto-minor-version-upgrade`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

Indica que as atualizações de mecanismos menores são aplicadas automaticamente à instância de banco de dados durante a janela de manutenção.

Padrão: `true`

- AvailabilityZone (na CLI: `--availability-zone`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  A zona de disponibilidade do EC2 na qual a instância de banco de dados é criada.

  Padrão: uma zona de disponibilidade aleatória e escolhida pelo sistema na região da Amazon do endpoint.

  Exemplo: `us-east-1d`

  Restrição: o parâmetro AvailabilityZone não poderá ser especificado se o parâmetro MultiAZ estiver definido como `true`. A zona de disponibilidade especificada deve estar na mesma região da Amazon que o endpoint atual.
- BackupRetentionPeriod (na CLI: `--backup-retention-period`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número de dias durante os quais os backups automatizados são retidos.

  Não aplicável. O período de retenção de backups automatizados é gerenciado pelo cluster de banco de dados. Para obter mais informações, consulte the section called “CreateDBCluster”.

  Padrão: 1

  Restrições:

  - Deve ser um valor de 0 a 35
  - Não poderá ser definido como 0 se a instância de banco de dados for uma origem para réplicas de leitura
- CopyTagsToSnapshot (na CLI: `--copy-tags-to-snapshot`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  True para copiar todas as tags da instância de banco de dados para os snapshots da instância de banco de dados. Caso contrário, false. O padrão é falso.
- DBClusterIdentifier (na CLI: `--db-cluster-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

O identificador do cluster de banco de dados ao qual a instância pertencerá.

Para obter informações sobre como criar com um cluster de banco de dados, consulte the section called “CreateDBCluster”.

Tipo: string

- DBInstanceClass (na CLI: `--db-instance-class`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  A capacidade de computação e memória da instância de banco de dados, por exemplo, `db.m4.large`. Nem todas as classes de instâncias de banco de dados estão disponíveis em todas as regiões da Amazon.
- DBInstanceIdentifier (na CLI: `--db-instance-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O DB instance identifier. Este parâmetro é armazenado como uma string com letras minúsculas.

  Restrições:

  - Deve conter de 1 a 63 caracteres, incluindo letras, números ou hífens.
  - O primeiro caractere deve ser uma letra.
  - Não podem terminar com um hífen ou conter dois hífens consecutivos.

  Exemplo: `mydbinstance`
- DBName (na CLI: `--db-name`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Sem suporte.
- DBParameterGroupName (na CLI: `--db-parameter-group-name`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do grupo de parâmetros de banco de dados a ser associado a essa instância de banco de dados. Se este argumento for omitido, o DBParameterGroup padrão para o mecanismo especificado será usado.

  Restrições:

  - Deve ter de 1 a 255 letras, números ou hifens.
  - O primeiro caractere deve ser uma letra
  - Não podem terminar com um hífen ou conter dois hífens consecutivos

- DBSecurityGroups (na CLI: `--db-security-groups`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de grupos de segurança de banco de dados a ser associada a essa instância de banco de dados.

  Padrão: o grupo de segurança de banco de dados padrão para o mecanismo de banco de dados.
- DBSubnetGroupName (na CLI: `--db-subnet-group-name`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um grupo de sub-redes de banco de dados a ser associado a essa instância de banco de dados.

  Se não houver um grupo de sub-redes de banco de dados, será uma instância de banco de dados que não seja da VPC.
- DeletionProtection (na CLI: `--deletion-protection`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Um valor que indica se a instância de banco de dados tem a proteção contra exclusão habilitada. O banco de dados não pode ser excluído quando a proteção contra exclusão está habilitada. Por padrão, a proteção contra exclusão fica desabilitada. Consulte Excluir uma instância de banco de dados.

  As instâncias de banco de dados em um cluster de banco de dados poderão ser excluídas mesmo quando a proteção contra exclusão estiver ativada no cluster de banco de dados pai.
- Domain (na CLI: `--domain`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o domínio do Active Directory no qual criar a instância.
- DomainIAMRoleName (na CLI: `--domain-iam-role-name`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifique o nome da função do IAM a ser usado ao fazer chamadas de API para o Directory Service.
- EnableCloudwatchLogsExports (na CLI: `--enable-cloudwatch-logs-exports`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  A lista de tipos de log que precisam estar habilitados para exportação para o CloudWatch Logs.
- EnableIAMDatabaseAuthentication (na CLI: `--enable-iam-database-authentication`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

Não compatível com o Neptune (ignorado).

- Engine (na CLI: `--engine`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do mecanismo de banco de dados a ser usado para essa instância.

  Valores Válidos: `neptune`
- EngineVersion (na CLI: `--engine-version`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O número da versão do mecanismo de banco de dados a ser usado. Atualmente, a definição desse parâmetro não tem efeito.
- Iops (na CLI: `--iops`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  A quantidade de IOPS (operações de entrada/saída por segundo) provisionadas a serem inicialmente alocadas para a instância de banco de dados.
- KmsKeyId (na CLI: `--kms-key-id`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador da chave do Amazon KMS para uma instância de banco de dados criptografada.

  O identificador de chave KMS é o Amazon Resource Name (ARN) da chave de criptografia KMS. Se você estiver criando uma instância de banco de dados com a mesma conta da Amazon que tem a chave de criptografia do KMS usada para criptografar a nova instância de banco de dados, use o alias da chave do KMS em vez do ARN para a chave de criptografia do KM.

  Não aplicável. O identificador de chave do KMS é gerenciado pelo cluster de banco de dados. Para obter mais informações, consulte the section called “CreateDBCluster”.

  Se o parâmetro `StorageEncrypted` for true e você não especificar o valor para o parâmetro `KmsKeyId`, o Amazon Neptune usará a chave de criptografia padrão. O Amazon KMS cria a chave de criptografia padrão para a conta da Amazon. A conta da Amazon tem uma chave de criptografia padrão diferente para cada região da Amazon.
- LicenseModel (na CLI: `--license-model`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Informações do modelo de licença dessa instância de banco de dados.

Valores válidos: `license-included` | `bring-your-own-license` | `general-public-license`

- MonitoringInterval (na CLI: `--monitoring-interval`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O intervalo, em segundos, entre pontos quando as métricas de monitoramento avançado são coletadas para a instância de banco de dados. Para desabilitar a coleta de métricas de monitoramento avançado, especifique 0. O padrão é 0.

  Se `MonitoringRoleArn` for especificado, você também deverá definir `MonitoringInterval` como um valor diferente de 0.

  Valores Válidos: `0, 1, 5, 10, 15, 30, 60`
- MonitoringRoleArn (na CLI: `--monitoring-role-arn`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN da função do IAM que permite que o Neptune envie métricas de monitoramento avançado para o Amazon CloudWatch Logs. Por exemplo, `arn:aws:iam:123456789012:role/emaccess`.

  Se `MonitoringInterval` estiver definido com um valor diferente de 0, você deverá fornecer um valor de `MonitoringRoleArn`.
- MultiAZ (na CLI: `--multi-az`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se a instância de banco de dados é uma implantação Multi-AZ. Não será possível definir o parâmetro AvailabilityZone se o parâmetro MultiAZ estiver definido como true.
- Port (na CLI: `--port`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número da porta na qual o banco de dados aceita conexões.

  Não aplicável. A porta é gerenciada pelo cluster de banco de dados. Para obter mais informações, consulte the section called “CreateDBCluster”.

  Padrão: 8182

  Tipo: inteiro

- PreferredBackupWindow (na CLI: `--preferred-backup-window`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O intervalo de tempo diário durante o qual os backups automatizados são criados.

  Não aplicável. O intervalo de tempo diário para a criação de backups automatizados é gerenciado pelo cluster de banco de dados. Para obter mais informações, consulte the section called “CreateDBCluster”.
- PreferredMaintenanceWindow (na CLI: `--preferred-maintenance-window`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O intervalo de tempo em cada semana durante o qual ocorre a manutenção do sistema, no Tempo Universal Coordenado (UTC).

  Formato: `ddd:hh24:mi-ddd:hh24:mi`

  O padrão é uma janela de trinta minutos selecionada aleatoriamente em um bloco de tempo de oito horas para cada região da Amazon, ocorrendo em um dia aleatório da semana.

  Dias válidos: Mon, Tue, Wed, Thu, Fri, Sat, Sun.

  Restrições: janela mínima de 30 minutos.
- PromotionTier (na CLI: `--promotion-tier`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Um valor que especifica a ordem na qual uma réplica de leitura é promovida à instância principal após uma falha da instância principal existente.

  Padrão: 1

  Valores válidos: 0 a 15
- PubliclyAccessible (na CLI: `--publicly-accessible`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Este sinalizador não deve mais ser usado.
- StorageEncrypted (na CLI: `--storage-encrypted`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se a instância de banco de dados é criptografada.

Não aplicável. A criptografia para instâncias de banco de dados é gerenciada pelo cluster de banco de dados. Para obter mais informações, consulte the section called “CreateDBCluster”.

Padrão: False

- StorageType (na CLI: `--storage-type`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Não aplicável. No Neptune, o tipo de armazenamento é gerenciado no nível do cluster de banco de dados.
- Tags (na CLI: `--tags`): uma matriz de objetos Tag.

  As tags a serem atribuídas à nova instância.
- TdeCredentialArn (na CLI: `--tde-credential-arn`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN do armazenamento de chaves ao qual associar a instância para a criptografia TDE.
- TdeCredentialPassword (na CLI: `--tde-credential-password`): uma SensitiveString, do tipo: `string` (uma string codificada em UTF-8).

  A senha para o determinado ARN do armazenamento de chaves para acessar o dispositivo.
- Timezone (na CLI: `--timezone`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O fuso horário da instância de banco de dados.
- VpcSecurityGroupIds (na CLI: `--vpc-security-group-ids`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de grupos de segurança da VPC do EC2 a ser associada a essa instância de banco de dados.

  Não aplicável. A lista de grupos de segurança da VPC do EC2 associada é gerenciada pelo cluster de banco de dados. Para obter mais informações, consulte the section called “CreateDBCluster”.

  Padrão: o grupo de segurança da VPC do EC2 padrão para a VPC do grupo de sub-redes de banco de dados.

Resposta

Contém os detalhes de uma instância de banco de dados do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta na ação the section called "DescribeDBInstances".

- AutoMinorVersionUpgrade: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica que os patches de versões secundárias são aplicados automaticamente.
- AvailabilityZone: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o nome da zona de disponibilidade na qual a instância de banco de dados está localizada.
- BackupRetentionPeriod: um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica o número de dias durante os quais os DB snapshots automáticos são retidos.
- CACertificateIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador do certificado da CA para essa instância DB.
- CopyTagsToSnapshot: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se as tags atribuídas são copiadas da instância de banco de dados nos snapshots da instância de banco de dados.
- DBClusterIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Se a instância de banco de dados for membro de um cluster de banco de dados, conterá o nome do cluster de banco de dados do qual a instância de banco de dados é membro.
- DBInstanceArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) da instância de banco de dados.
- DBInstanceClass: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém o nome da classe de capacidade de computação e memória da instância de banco de dados.
- DBInstanceIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um identificador de banco de dados fornecido pelo usuário. Esse identificador é a chave exclusiva que identifica uma instância de banco de dados.
- DbInstancePort: um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

Especifica a porta na qual a instância de banco de dados escuta. Se a instância de banco de dados fizer parte de um cluster de banco de dados, isso poderá ser uma porta diferente da porta do cluster de banco de dados.

- DBInstanceStatus: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o estado atual desse banco de dados.

- DbiResourceId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador exclusivo e imutável da região da Amazon da instância de banco de dados. Esse identificador é encontrado nas entradas de log do Amazon CloudTrail sempre que a chave do Amazon KMS da instância de banco de dados é acessada.

- DBName: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do banco de dados.

- DBParameterGroups – Uma matriz de objetos DBParameterGroupStatus.

  Fornece a lista de grupo de parâmetros de banco de dados aplicados a essa instância de banco de dados.

- DBSecurityGroups – Uma matriz de objetos DBSecurityGroupMembership.

  Fornece a lista de elementos do grupo de segurança de banco de dados que contém somente os subelementos `DBSecurityGroup.Name` e `DBSecurityGroup.Status`.

- DBSubnetGroup: um objeto DBSubnetGroup.

  Especifica informações sobre o grupo de sub-redes associado à instância de banco de dados, incluindo o nome, a descrição e as sub-redes no grupo de sub-redes.

- DeletionProtection: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica se a instância de banco de dados tem a proteção contra exclusão ativada. A instância não poderá ser excluída quando a proteção contra exclusão estiver ativada. Consulte Excluir uma instância de banco de dados.

- DomainMemberships – Uma matriz de objetos DomainMembership.

  Não suportado

- EnabledCloudwatchLogsExports: uma string, do tipo: `string` (uma string codificada em UTF-8).

Uma lista de tipos de logs que essa instância de banco de dados está configurada para exportar para o CloudWatch Logs.

- Endpoint – Um objeto Endpoint.

  Especifica o endpoint de conexão.

- Engine: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o nome do mecanismo de banco de dados a ser usado para essa instância de banco de dados.

- EngineVersion: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Indica a versão do mecanismo do banco de dados.

- EnhancedMonitoringResourceArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) do fluxo de logs do Amazon CloudWatch Logs que recebe os dados de métricas de monitoramento avançado para a instância de banco de dados.

- IAMDatabaseAuthenticationEnabled: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Verdadeiro se a autenticação do Amazon Identity and Access Management (IAM) estiver habilitada. Caso contrário, falso.

- InstanceCreateTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Fornece a data e hora em que a instância de banco de dados foi criada.

- Iops: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica o valor das IOPS (operações de entrada/saída por segundo) provisionadas.

- KmsKeyId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Não compatível: a criptografia para instâncias de banco de dados é gerenciada pelo cluster de banco de dados.

- LatestRestorableTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a data mais recente até a qual um banco de dados pode ser restaurado com restauração pontual.

- LicenseModel: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Informações do modelo de licença dessa instância de banco de dados.
- MonitoringInterval: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O intervalo, em segundos, entre pontos quando as métricas de monitoramento avançado são coletadas para a instância de banco de dados.
- MonitoringRoleArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN da função do IAM que permite que o Neptune envie métricas de monitoramento avançado para o Amazon CloudWatch Logs.
- MultiAZ: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se a instância de banco de dados é uma implantação Multi-AZ.
- PendingModifiedValues: um objeto PendingModifiedValues.

  Especifica que alterações à instância de banco de dados estão pendentes. Esse elemento só é incluído quando as alterações estão pendentes. As alterações específicas são identificadas por subelementos.
- PreferredBackupWindow: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o intervalo de tempo diário durante o qual os backups automatizados serão criados se eles estiverem habilitados, conforme determinado por `BackupRetentionPeriod`.
- PreferredMaintenanceWindow: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o período semanal durante o qual pode ocorrer a manutenção do sistema, em Tempo Universal Coordenado (UTC).
- PromotionTier: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Um valor que especifica a ordem em que uma réplica de leitura é promovida à instância principal após uma falha da instância principal existente.
- PubliclyAccessible: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Este sinalizador não deve mais ser usado.
- ReadReplicaDBClusterIdentifiers: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um ou mais identificadores de clusters de banco de dados que são réplicas de leitura dessa instância de banco de dados.

- ReadReplicaDBInstanceIdentifiers: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um ou mais identificadores das réplicas de leitura associadas a essa instância de banco de dados.
- ReadReplicaSourceDBInstanceIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém o identificador da instância de banco de dados de origem se essa instância de banco de dados for uma réplica de leitura.
- SecondaryAvailabilityZone: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Se houver, especificará o nome da zona de disponibilidade secundária para uma instância de banco de dados com suporte a Multi-AZ
- StatusInfos – Uma matriz de objetos DBInstanceStatusInfo.

  O status da réplica de leitura. Se a instância não for uma réplica de leitura, isso ficará em branco.
- StorageEncrypted: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Não compatível: a criptografia para instâncias de banco de dados é gerenciada pelo cluster de banco de dados.
- StorageType: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o tipo de armazenamento associado à instância de banco de dados.
- TdeCredentialArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN do armazenamento de chaves ao qual a instância está associada para criptografia TDE.
- Timezone: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Sem suporte.
- VpcSecurityGroups – Uma matriz de objetos VpcSecurityGroupMembership.

  Fornece uma lista de elementos do grupo de segurança da VPC à qual a instância de banco de dados pertence.

Erros

- DBInstanceAlreadyExistsFault
- InsufficientDBInstanceCapacityFault

- DBParameterGroupNotFoundFault
- DBSecurityGroupNotFoundFault
- InstanceQuotaExceededFault
- StorageQuotaExceededFault
- DBSubnetGroupNotFoundFault
- DBSubnetGroupDoesNotCoverEnoughAZs
- InvalidDBClusterStateFault
- InvalidSubnet
- InvalidVPCNetworkStateFault
- ProvisionedIopsNotAvailableInAZFault
- OptionGroupNotFoundFault
- DBClusterNotFoundFault
- StorageTypeNotSupportedFault
- AuthorizationNotFoundFault
- KMSKeyNotAccessibleFault
- DomainNotFoundFault

DeleteDBInstance (ação)

O nome da CLI da AWS para essa API é: `delete-db-instance`.

A ação DeleteDBInstance exclui uma instância de banco de dados provisionada anteriormente. Quando você exclui uma instância de banco de dados, todos os backups automatizados dessa instância são excluídos e não podem ser recuperados. Os DB snapshots manuais da instância de banco de dados a ser excluída por `DeleteDBInstance` não serão excluídos.

Se você solicitar um DB snapshot final, o status da instância de banco de dados do Amazon Neptune será `deleting` até que o DB snapshot seja criado. A ação de API `DescribeDBInstance` é usada para monitorar o status dessa operação. A ação não poderá ser cancelada nem revertida após ser enviada.

Observe que, quando uma instância de banco de dados estiver em um estado de falha e tiver um status `failed`, `incompatible-restore` ou `incompatible-network`, ela só poderá ser excluída quando o parâmetro `SkipFinalSnapshot` for definido como `true`.

Não será possível excluir uma instância de banco de dados se ela for a única instância no cluster de banco de dados ou se ela tiver proteção contra exclusão ativada.

Solicitação

- DBInstanceIdentifier (na CLI: `--db-instance-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador da instância de banco de dados a ser excluída. Este parâmetro não diferencia maiúsculas de minúsculas.

  Restrições:

  - Deve corresponder ao nome de uma instância de banco de dados existente.
- FinalDBSnapshotIdentifier (na CLI: `--final-db-snapshot-identifier`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O DBSnapshotIdentifier do novo DBSnapshot criado quando SkipFinalSnapshot está definido como `false`.

  > **Note**
  >
  > Especificar esse parâmetro e também definir o parâmetro SkipFinalShapshot como true resultará em erro.

  Restrições:

  - Deve ter de 1 a 255 letras ou números.
  - O primeiro caractere deve ser uma letra
  - Não podem terminar com um hífen ou conter dois hífens consecutivos
  - Não pode ser especificado ao excluir uma réplica de leitura.
- SkipFinalSnapshot (na CLI: `--skip-final-snapshot`): um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Determina se um DB snapshot final foi criado antes de a instância de banco de dados ser excluída. Se `true` for especificado, nenhum DBSnapshot será criado. Se `false` for especificado, um DB snapshot será criado antes de a instância de banco de dados ser excluída.

Observe que, quando uma instância de banco de dados estiver em um estado de falha e tiver um status "failed", "incompatible-restore" ou "incompatible-network", ela só poderá ser excluída quando o parâmetro SkipFinalSnapshot for definido como "true".

Especifique `true` ao excluir uma réplica de leitura.

> **Note**
>
> O parâmetro FinalDBSnapshotIdentifier deverá ser especificado se SkipFinalSnapshot for `false`.

Padrão: `false`

Resposta

Contém os detalhes de uma instância de banco de dados do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta na ação the section called "DescribeDBInstances".

- AutoMinorVersionUpgrade: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica que os patches de versões secundárias são aplicados automaticamente.
- AvailabilityZone: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o nome da zona de disponibilidade na qual a instância de banco de dados está localizada.
- BackupRetentionPeriod: um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica o número de dias durante os quais os DB snapshots automáticos são retidos.
- CACertificateIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador do certificado da CA para essa instância DB.
- CopyTagsToSnapshot: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

Especifica se as tags atribuídas são copiadas da instância de banco de dados nos snapshots da instância de banco de dados.

- DBClusterIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Se a instância de banco de dados for membro de um cluster de banco de dados, conterá o nome do cluster de banco de dados do qual a instância de banco de dados é membro.
- DBInstanceArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) da instância de banco de dados.
- DBInstanceClass: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém o nome da classe de capacidade de computação e memória da instância de banco de dados.
- DBInstanceIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um identificador de banco de dados fornecido pelo usuário. Esse identificador é a chave exclusiva que identifica uma instância de banco de dados.
- DbInstancePort: um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica a porta na qual a instância de banco de dados escuta. Se a instância de banco de dados fizer parte de um cluster de banco de dados, isso poderá ser uma porta diferente da porta do cluster de banco de dados.
- DBInstanceStatus: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o estado atual desse banco de dados.
- DbiResourceId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador exclusivo e imutável da região da Amazon da instância de banco de dados. Esse identificador é encontrado nas entradas de log do Amazon CloudTrail sempre que a chave do Amazon KMS da instância de banco de dados é acessada.
- DBName: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do banco de dados.
- DBParameterGroups – Uma matriz de objetos DBParameterGroupStatus.

  Fornece a lista de grupo de parâmetros de banco de dados aplicados a essa instância de banco de dados.

- DBSecurityGroups – Uma matriz de objetos DBSecurityGroupMembership.

  Fornece a lista de elementos do grupo de segurança de banco de dados que contém somente os subelementos `DBSecurityGroup.Name` e `DBSecurityGroup.Status`.

- DBSubnetGroup: um objeto DBSubnetGroup.

  Especifica informações sobre o grupo de sub-redes associado à instância de banco de dados, incluindo o nome, a descrição e as sub-redes no grupo de sub-redes.

- DeletionProtection: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica se a instância de banco de dados tem a proteção contra exclusão ativada. A instância não poderá ser excluída quando a proteção contra exclusão estiver ativada. Consulte Excluir uma instância de banco de dados.

- DomainMemberships – Uma matriz de objetos DomainMembership.

  Não suportado

- EnabledCloudwatchLogsExports: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de tipos de logs que essa instância de banco de dados está configurada para exportar para o CloudWatch Logs.

- Endpoint – Um objeto Endpoint.

  Especifica o endpoint de conexão.

- Engine: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o nome do mecanismo de banco de dados a ser usado para essa instância de banco de dados.

- EngineVersion: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Indica a versão do mecanismo do banco de dados.

- EnhancedMonitoringResourceArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) do fluxo de logs do Amazon CloudWatch Logs que recebe os dados de métricas de monitoramento avançado para a instância de banco de dados.

- IAMDatabaseAuthenticationEnabled: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

Verdadeiro se a autenticação do Amazon Identity and Access Management (IAM) estiver habilitada. Caso contrário, falso.

- InstanceCreateTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Fornece a data e hora em que a instância de banco de dados foi criada.

- Iops: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica o valor das IOPS (operações de entrada/saída por segundo) provisionadas.

- KmsKeyId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Não compatível: a criptografia para instâncias de banco de dados é gerenciada pelo cluster de banco de dados.

- LatestRestorableTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a data mais recente até a qual um banco de dados pode ser restaurado com restauração pontual.

- LicenseModel: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Informações do modelo de licença dessa instância de banco de dados.

- MonitoringInterval: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O intervalo, em segundos, entre pontos quando as métricas de monitoramento avançado são coletadas para a instância de banco de dados.

- MonitoringRoleArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN da função do IAM que permite que o Neptune envie métricas de monitoramento avançado para o Amazon CloudWatch Logs.

- MultiAZ: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se a instância de banco de dados é uma implantação Multi-AZ.

- PendingModifiedValues: um objeto PendingModifiedValues.

  Especifica que alterações à instância de banco de dados estão pendentes. Esse elemento só é incluído quando as alterações estão pendentes. As alterações específicas são identificadas por subelementos.

- PreferredBackupWindow: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o intervalo de tempo diário durante o qual os backups automatizados serão criados se eles estiverem habilitados, conforme determinado por `BackupRetentionPeriod`.
- PreferredMaintenanceWindow: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o período semanal durante o qual pode ocorrer a manutenção do sistema, em Tempo Universal Coordenado (UTC).
- PromotionTier: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Um valor que especifica a ordem em que uma réplica de leitura é promovida à instância principal após uma falha da instância principal existente.
- PubliclyAccessible: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Este sinalizador não deve mais ser usado.
- ReadReplicaDBClusterIdentifiers: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um ou mais identificadores de clusters de banco de dados que são réplicas de leitura dessa instância de banco de dados.
- ReadReplicaDBInstanceIdentifiers: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um ou mais identificadores das réplicas de leitura associadas a essa instância de banco de dados.
- ReadReplicaSourceDBInstanceIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém o identificador da instância de banco de dados de origem se essa instância de banco de dados for uma réplica de leitura.
- SecondaryAvailabilityZone: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Se houver, especificará o nome da zona de disponibilidade secundária para uma instância de banco de dados com suporte a Multi-AZ
- StatusInfos – Uma matriz de objetos DBInstanceStatusInfo.

  O status da réplica de leitura. Se a instância não for uma réplica de leitura, isso ficará em branco.
- StorageEncrypted: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

Não compatível: a criptografia para instâncias de banco de dados é gerenciada pelo cluster de banco de dados.

- StorageType: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o tipo de armazenamento associado à instância de banco de dados.

- TdeCredentialArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN do armazenamento de chaves ao qual a instância está associada para criptografia TDE.

- Timezone: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Sem suporte.

- VpcSecurityGroups – Uma matriz de objetos VpcSecurityGroupMembership.

  Fornece uma lista de elementos do grupo de segurança da VPC à qual a instância de banco de dados pertence.

Erros

- DBInstanceNotFoundFault
- InvalidDBInstanceStateFault
- DBSnapshotAlreadyExistsFault
- SnapshotQuotaExceededFault
- InvalidDBClusterStateFault

ModifyDBInstance (ação)

O nome da CLI da AWS para essa API é: `modify-db-instance`.

Modifica as configurações de uma instância de banco de dados. Você pode alterar um ou mais parâmetros de configuração do banco de dados especificando esses parâmetros e os novos valores na solicitação. Para saber quais modificações que você pode fazer em sua instância de banco de dados, chame the section called “DescribeValidDBInstanceModifications” antes de chamar the section called “ModifyDBInstance”.

Solicitação

- AllowMajorVersionUpgrade (na CLI: `--allow-major-version-upgrade`): um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica as atualizações principais de versões que são permitidas. Alterar esse parâmetro não resultará em uma interrupção e a alteração será aplicada de maneira assíncrona logo que possível.

- ApplyImmediately (na CLI: `--apply-immediately`): um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se as modificações feitas nessa solicitação e todas as modificações pendentes serão aplicadas de maneira assíncrona logo que possível, independentemente da configuração `PreferredMaintenanceWindow` da instância de banco de dados.

  Caso esse parâmetro seja definido como `false`, as alterações feitas na instância de banco de dados serão aplicadas durante a próxima janela de manutenção. Algumas alterações de parâmetro podem causar uma interrupção e serão aplicadas na próxima chamada para the section called “RebootDBInstance” ou na próxima reinicialização por falha.

  Padrão: `false`

- AutoMinorVersionUpgrade (na CLI: `--auto-minor-version-upgrade`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica que as atualizações secundárias do mecanismos serão aplicadas automaticamente à instância de banco de dados durante a janela de manutenção. Alterar esse parâmetro não resultará em uma interrupção, exceto no caso a seguir, e a alteração será aplicada de maneira assíncrona logo que possível. Ocorrerá uma interrupção se esse parâmetro estiver definido como `true` durante a janela de manutenção, e uma versão secundária mais nova estará disponível. O Neptune habilitou a aplicação automática de patches para essa versão do mecanismo.

- BackupRetentionPeriod (na CLI: `--backup-retention-period`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Não aplicável. O período de retenção de backups automatizados é gerenciado pelo cluster de banco de dados. Para obter mais informações, consulte the section called “ModifyDBCluster”.

  Padrão: usa a configuração existente

- CACertificateIdentifier (na CLI: `--ca-certificate-identifier`): uma string, do tipo: `string` (uma string codificada em UTF-8).

Indica o certificado que deve ser associado à instância.

- CloudwatchLogsExportConfiguration (na CLI: `--cloudwatch-logs-export-configuration`): um objeto CloudwatchLogsExportConfiguration.

  A configuração para os tipos de log a serem habilitados para exportação para o CloudWatch Logs para um determinado cluster ou instância de banco de dados.

- CopyTagsToSnapshot (na CLI: `--copy-tags-to-snapshot`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  True para copiar todas as tags da instância de banco de dados para os snapshots da instância de banco de dados. Caso contrário, false. O padrão é falso.

- DBInstanceClass (na CLI: `--db-instance-class`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  A nova capacidade de computação e memória da instância de banco de dados, por exemplo, `db.m4.large`. Nem todas as classes de instâncias de banco de dados estão disponíveis em todas as regiões da Amazon.

  Se você modificar a classe da instância de banco de dados, ocorrerá uma interrupção durante a alteração. A alteração será aplicada durante a próxima janela de manutenção, a menos que `ApplyImmediately` seja especificado como `true` para essa solicitação.

  Padrão: usa a configuração existente

- DBInstanceIdentifier (na CLI: `--db-instance-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O DB instance identifier. Esse valor é armazenado como uma string em minúsculas.

  Restrições:

  - Deve corresponder ao identificador de uma DBInstance existente.

- DBParameterGroupName (na CLI: `--db-parameter-group-name`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do grupo de parâmetros de banco de dados a ser aplicado à instância de banco de dados. Alterar essa configuração não resultará em uma interrupção. O nome do parameter group propriamente dito é alterado imediatamente, mas as alterações de parâmetros reais não serão aplicadas até que você reinicialize a instância sem failover. A instância de banco de dados NÃO

será reinicializada automaticamente, e as alterações de parâmetro NÃO serão aplicadas durante a próxima janela de manutenção.

Padrão: usa a configuração existente

Restrições: o grupo de parâmetros de banco de dados deve estar na mesma família do grupo de parâmetros de banco de dados como essa instância de banco de dados.

- DBPortNumber (na CLI: `--db-port-number`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número da porta na qual o banco de dados aceita conexões.

  O valor do parâmetro `DBPortNumber` não deve corresponder a nenhum dos valores de porta especificados para as opções no grupo de opções da instância de banco de dados.

  Seu banco de dados será reiniciado quando você alterar o valor de `DBPortNumber` independentemente do valor do parâmetro `ApplyImmediately`.

  Padrão: `8182`

- DBSecurityGroups (na CLI: `--db-security-groups`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de grupos de segurança de banco de dados a serem autorizados nessa instância de banco de dados. A alteração dessa configuração não resultará em uma interrupção e a alteração será aplicada de maneira assíncrona logo que possível.

  Restrições:

  - Se fornecido, deve corresponder a DBSecurityGroups existentes.

- DBSubnetGroupName (na CLI: `--db-subnet-group-name`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O novo grupo de sub-redes de banco de dados para a instância de banco de dados. Você pode usar esse parâmetro para mover sua instância de banco de dados para uma VPC diferente.

  Alterar o grupo de sub-redes faz com que ocorra uma interrupção durante a alteração. A alteração será aplicada durante a próxima janela de manutenção, a menos que você especifique `true` para o parâmetro `ApplyImmediately`.

  Restrições: se fornecido, deve corresponder ao nome de um DBSubnetGroup existente.

Exemplo: `mySubnetGroup`

- DeletionProtection (na CLI: `--deletion-protection`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Um valor que indica se a instância de banco de dados tem a proteção contra exclusão habilitada. O banco de dados não pode ser excluído quando a proteção contra exclusão está habilitada. Por padrão, a proteção contra exclusão fica desabilitada. Consulte Excluir uma instância de banco de dados.

- Domain (na CLI: `--domain`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Sem suporte.

- DomainIAMRoleName (na CLI: `--domain-iam-role-name`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Não suportado

- EnableIAMDatabaseAuthentication (na CLI: `--enable-iam-database-authentication`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Verdadeiro para habilitar o mapeamento de contas do Amazon Identity and Access Management (IAM) para contas de banco de dados. Caso contrário, falso.

  Você pode habilitar a autenticação do banco de dados do IAM para os mecanismos de banco de dados a seguir

  Não aplicável. O mapeamento de contas do Amazon IAM para contas de banco de dados é gerenciado pelo cluster de banco de dados. Para obter mais informações, consulte the section called “ModifyDBCluster”.

  Padrão: `false`

- EngineVersion (na CLI: `--engine-version`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O número da versão do mecanismo de banco de dados para a qual será feita a atualização. Atualmente, a definição desse parâmetro não tem efeito. Para atualizar seu mecanismo de banco de dados para a versão mais recente, use a API the section called “ApplyPendingMaintenanceAction”.

- Iops (na CLI: `--iops`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O novo valor de IOPS (operações de E/S por segundo) provisionadas para a instância.

  Alterar essa configuração parâmetro não resultará em uma interrupção e a alteração será aplicada durante a próxima janela de manutenção, a menos que o parâmetro `ApplyImmediately` esteja definido como `true` para essa solicitação.

  Padrão: usa a configuração existente
- MonitoringInterval (na CLI: `--monitoring-interval`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O intervalo, em segundos, entre pontos quando as métricas de monitoramento avançado são coletadas para a instância de banco de dados. Para desabilitar a coleta de métricas de monitoramento avançado, especifique 0. O padrão é 0.

  Se `MonitoringRoleArn` for especificado, você também deverá definir `MonitoringInterval` como um valor diferente de 0.

  Valores Válidos: `0, 1, 5, 10, 15, 30, 60`
- MonitoringRoleArn (na CLI: `--monitoring-role-arn`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN da função do IAM que permite que o Neptune envie métricas de monitoramento avançado para o Amazon CloudWatch Logs. Por exemplo, `arn:aws:iam:123456789012:role/emaccess`.

  Se `MonitoringInterval` estiver definido com um valor diferente de 0, você deverá fornecer um valor de `MonitoringRoleArn`.
- MultiAZ (na CLI: `--multi-az`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se a instância de banco de dados é uma implantação Multi-AZ. Alterar esse parâmetro não resultará em uma interrupção e a alteração será aplicada durante a próxima janela de manutenção, a menos que o parâmetro `ApplyImmediately` esteja definido como `true` para essa solicitação.
- NewDBInstanceIdentifier (na CLI: `--new-db-instance-identifier`): uma string, do tipo: `string` (uma string codificada em UTF-8).

O novo identificador da instância de banco de dados durante a renomeação de uma instância de banco de dados. Quando você altera o identificador da instância de banco de dados, uma reinicialização de instância ocorrerá imediatamente se você definir `Apply Immediately` como true ou ocorrerá durante a próxima janela de manutenção se você definir `Apply Immediately` como false. Esse valor é armazenado como uma string em minúsculas.

Restrições:

- Deve conter de 1 a 63 caracteres, incluindo letras, números ou hífens.
- O primeiro caractere deve ser uma letra.
- Não podem terminar com um hífen ou conter dois hífens consecutivos.

Exemplo: `mydbinstance`

- PreferredBackupWindow (na CLI: `--preferred-backup-window`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O período diário durante o qual os backups automatizados serão criados se eles estiverem habilitados.

  Não aplicável. O intervalo de tempo diário para a criação de backups automatizados é gerenciado pelo cluster de banco de dados. Para obter mais informações, consulte the section called “ModifyDBCluster”.

  Restrições:

  - Deve estar no formato hh24:mi-hh24:mi
  - Deve estar expresso no Tempo Universal Coordenado (UTC)
  - Não pode entrar em conflito com a janela de manutenção preferencial
  - Deve ser, pelo menos, 30 minutos
- PreferredMaintenanceWindow (na CLI: `--preferred-maintenance-window`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O período semanal (em UTC) durante o qual pode ocorrer a manutenção do sistema, o que pode resultar em uma interrupção. A alteração desse parâmetro não resultará em uma interrupção, exceto na situação a seguir, e a alteração será aplicada de maneira assíncrona logo que possível. Se houver ações pendentes que causem uma reinicialização e a janela de manutenção for alterada para incluir a hora atual, a alteração desse parâmetro causará uma reinicialização da instância de banco de dados. Se essa janela for mudada para a hora atual, deverá haver pelo

menos 30 minutos entre a hora atual e o final da janela para garantir que as alterações pendentes sejam aplicadas.

Padrão: usa a configuração existente

Formato: ddd:hh24:mi-ddd:hh24:mi

Dias válidos: Mon | Tue | Wed | Thu | Fri | Sat | Sun

Restrições: deve ser, pelo menos, 30 minutos

- PromotionTier (na CLI: `--promotion-tier`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Um valor que especifica a ordem em que uma réplica de leitura é promovida à instância principal após uma falha da instância principal existente.

  Padrão: 1

  Valores válidos: 0 a 15

- PubliclyAccessible (na CLI: `--publicly-accessible`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Este sinalizador não deve mais ser usado.

- StorageType (na CLI: `--storage-type`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Não aplicável. No Neptune, o tipo de armazenamento é gerenciado no nível do cluster de banco de dados.

- TdeCredentialArn (na CLI: `--tde-credential-arn`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN do armazenamento de chaves ao qual associar a instância para a criptografia TDE.

- TdeCredentialPassword (na CLI: `--tde-credential-password`): uma SensitiveString, do tipo: `string` (uma string codificada em UTF-8).

  A senha para o determinado ARN do armazenamento de chaves para acessar o dispositivo.

- VpcSecurityGroupIds (na CLI: `--vpc-security-group-ids`): uma string, do tipo: `string` (uma string codificada em UTF-8).

Uma lista de grupos de segurança da VPC do EC2 a serem autorizados nessa instância de banco de dados. Essa alteração será aplicada de forma assíncrona logo que possível.

Não aplicável. A lista de grupos de segurança da VPC do EC2 associada é gerenciada pelo cluster de banco de dados. Para obter mais informações, consulte the section called “ModifyDBCluster”.

Restrições:

- Se fornecido, deve corresponder a VpcSecurityGroupIds existentes.

Resposta

Contém os detalhes de uma instância de banco de dados do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta na ação the section called “DescribeDBInstances”.

- AutoMinorVersionUpgrade: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica que os patches de versões secundárias são aplicados automaticamente.
- AvailabilityZone: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o nome da zona de disponibilidade na qual a instância de banco de dados está localizada.
- BackupRetentionPeriod: um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica o número de dias durante os quais os DB snapshots automáticos são retidos.
- CACertificateIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador do certificado da CA para essa instância DB.
- CopyTagsToSnapshot: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se as tags atribuídas são copiadas da instância de banco de dados nos snapshots da instância de banco de dados.
- DBClusterIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

Se a instância de banco de dados for membro de um cluster de banco de dados, conterá o nome do cluster de banco de dados do qual a instância de banco de dados é membro.

- DBInstanceArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) da instância de banco de dados.

- DBInstanceClass: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém o nome da classe de capacidade de computação e memória da instância de banco de dados.

- DBInstanceIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um identificador de banco de dados fornecido pelo usuário. Esse identificador é a chave exclusiva que identifica uma instância de banco de dados.

- DbInstancePort: um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica a porta na qual a instância de banco de dados escuta. Se a instância de banco de dados fizer parte de um cluster de banco de dados, isso poderá ser uma porta diferente da porta do cluster de banco de dados.

- DBInstanceStatus: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o estado atual desse banco de dados.

- DbiResourceId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador exclusivo e imutável da região da Amazon da instância de banco de dados. Esse identificador é encontrado nas entradas de log do Amazon CloudTrail sempre que a chave do Amazon KMS da instância de banco de dados é acessada.

- DBName: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do banco de dados.

- DBParameterGroups – Uma matriz de objetos DBParameterGroupStatus.

  Fornece a lista de grupo de parâmetros de banco de dados aplicados a essa instância de banco de dados.

- DBSecurityGroups – Uma matriz de objetos DBSecurityGroupMembership.

  Fornece a lista de elementos do grupo de segurança de banco de dados que contém somente os subelementos `DBSecurityGroup.Name` e `DBSecurityGroup.Status`.

- DBSubnetGroup: um objeto DBSubnetGroup.

  Especifica informações sobre o grupo de sub-redes associado à instância de banco de dados, incluindo o nome, a descrição e as sub-redes no grupo de sub-redes.
- DeletionProtection: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica se a instância de banco de dados tem a proteção contra exclusão ativada. A instância não poderá ser excluída quando a proteção contra exclusão estiver ativada. Consulte Excluir uma instância de banco de dados.
- DomainMemberships – Uma matriz de objetos DomainMembership.

  Não suportado
- EnabledCloudwatchLogsExports: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de tipos de logs que essa instância de banco de dados está configurada para exportar para o CloudWatch Logs.
- Endpoint – Um objeto Endpoint.

  Especifica o endpoint de conexão.
- Engine: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o nome do mecanismo de banco de dados a ser usado para essa instância de banco de dados.
- EngineVersion: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Indica a versão do mecanismo do banco de dados.
- EnhancedMonitoringResourceArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) do fluxo de logs do Amazon CloudWatch Logs que recebe os dados de métricas de monitoramento avançado para a instância de banco de dados.
- IAMDatabaseAuthenticationEnabled: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Verdadeiro se a autenticação do Amazon Identity and Access Management (IAM) estiver habilitada. Caso contrário, falso.
- InstanceCreateTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

Fornece a data e hora em que a instância de banco de dados foi criada.

- Iops: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica o valor das IOPS (operações de entrada/saída por segundo) provisionadas.
- KmsKeyId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Não compatível: a criptografia para instâncias de banco de dados é gerenciada pelo cluster de banco de dados.
- LatestRestorableTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a data mais recente até a qual um banco de dados pode ser restaurado com restauração pontual.
- LicenseModel: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Informações do modelo de licença dessa instância de banco de dados.
- MonitoringInterval: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O intervalo, em segundos, entre pontos quando as métricas de monitoramento avançado são coletadas para a instância de banco de dados.
- MonitoringRoleArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN da função do IAM que permite que o Neptune envie métricas de monitoramento avançado para o Amazon CloudWatch Logs.
- MultiAZ: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se a instância de banco de dados é uma implantação Multi-AZ.
- PendingModifiedValues: um objeto PendingModifiedValues.

  Especifica que alterações à instância de banco de dados estão pendentes. Esse elemento só é incluído quando as alterações estão pendentes. As alterações específicas são identificadas por subelementos.
- PreferredBackupWindow: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o intervalo de tempo diário durante o qual os backups automatizados serão criados se eles estiverem habilitados, conforme determinado por `BackupRetentionPeriod`.
- PreferredMaintenanceWindow: uma string, do tipo: `string` (uma string codificada em UTF-8).

Especifica o período semanal durante o qual pode ocorrer a manutenção do sistema, em Tempo Universal Coordenado (UTC).

- PromotionTier: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Um valor que especifica a ordem em que uma réplica de leitura é promovida à instância principal após uma falha da instância principal existente.
- PubliclyAccessible: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Este sinalizador não deve mais ser usado.
- ReadReplicaDBClusterIdentifiers: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um ou mais identificadores de clusters de banco de dados que são réplicas de leitura dessa instância de banco de dados.
- ReadReplicaDBInstanceIdentifiers: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um ou mais identificadores das réplicas de leitura associadas a essa instância de banco de dados.
- ReadReplicaSourceDBInstanceIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém o identificador da instância de banco de dados de origem se essa instância de banco de dados for uma réplica de leitura.
- SecondaryAvailabilityZone: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Se houver, especificará o nome da zona de disponibilidade secundária para uma instância de banco de dados com suporte a Multi-AZ
- StatusInfos – Uma matriz de objetos DBInstanceStatusInfo.

  O status da réplica de leitura. Se a instância não for uma réplica de leitura, isso ficará em branco.
- StorageEncrypted: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Não compatível: a criptografia para instâncias de banco de dados é gerenciada pelo cluster de banco de dados.
- StorageType: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o tipo de armazenamento associado à instância de banco de dados.
- TdeCredentialArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

O ARN do armazenamento de chaves ao qual a instância está associada para criptografia TDE.

- Timezone: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Sem suporte.

- VpcSecurityGroups – Uma matriz de objetos VpcSecurityGroupMembership.

  Fornece uma lista de elementos do grupo de segurança da VPC à qual a instância de banco de dados pertence.

Erros

- InvalidDBInstanceStateFault
- InvalidDBSecurityGroupStateFault
- DBInstanceAlreadyExistsFault
- DBInstanceNotFoundFault
- DBSecurityGroupNotFoundFault
- DBParameterGroupNotFoundFault
- InsufficientDBInstanceCapacityFault
- StorageQuotaExceededFault
- InvalidVPCNetworkStateFault
- ProvisionedIopsNotAvailableInAZFault
- OptionGroupNotFoundFault
- DBUpgradeDependencyFailureFault
- StorageTypeNotSupportedFault
- AuthorizationNotFoundFault
- CertificateNotFoundFault
- DomainNotFoundFault

RebootDBInstance (ação)

O nome da CLI da AWS para essa API é: `reboot-db-instance`.

Você pode precisar reinicializar sua instância de banco de dados, geralmente, por motivos de manutenção. Por exemplo, se você fizer determinadas modificações ou alterar o grupo de parâmetros de banco de dados associado à instância de banco de dados, deverá reiniciar a instância para que as alterações sejam implementadas.

Reiniciar uma instância de banco de dados reinicia o serviço de mecanismo de banco de dados. Reinicializar uma instância de banco de dados resulta em uma interrupção momentânea, durante a qual o status da instância de banco de dados é definido como rebooting (reinicialização).

Solicitação

- DBInstanceIdentifier (na CLI: `--db-instance-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O DB instance identifier. Este parâmetro é armazenado como uma string com letras minúsculas.

  Restrições:

  - Deve corresponder ao identificador de uma DBInstance existente.
- ForceFailover (na CLI: `--force-failover`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Quando `true`, a reinicialização é feita por meio de um failover de Multi-AZ.

  Restrição: não será possível especificar `true` se a instância não estiver configurada para Multi-AZ.

Resposta

Contém os detalhes de uma instância de banco de dados do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta na ação the section called “DescribeDBInstances”.

- AutoMinorVersionUpgrade: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica que os patches de versões secundárias são aplicados automaticamente.
- AvailabilityZone: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o nome da zona de disponibilidade na qual a instância de banco de dados está localizada.

- BackupRetentionPeriod: um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica o número de dias durante os quais os DB snapshots automáticos são retidos.

- CACertificateIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador do certificado da CA para essa instância DB.

- CopyTagsToSnapshot: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se as tags atribuídas são copiadas da instância de banco de dados nos snapshots da instância de banco de dados.

- DBClusterIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Se a instância de banco de dados for membro de um cluster de banco de dados, conterá o nome do cluster de banco de dados do qual a instância de banco de dados é membro.

- DBInstanceArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) da instância de banco de dados.

- DBInstanceClass: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém o nome da classe de capacidade de computação e memória da instância de banco de dados.

- DBInstanceIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um identificador de banco de dados fornecido pelo usuário. Esse identificador é a chave exclusiva que identifica uma instância de banco de dados.

- DbInstancePort: um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica a porta na qual a instância de banco de dados escuta. Se a instância de banco de dados fizer parte de um cluster de banco de dados, isso poderá ser uma porta diferente da porta do cluster de banco de dados.

- DBInstanceStatus: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o estado atual desse banco de dados.

- DbiResourceId: uma string, do tipo: `string` (uma string codificada em UTF-8).

O identificador exclusivo e imutável da região da Amazon da instância de banco de dados. Esse identificador é encontrado nas entradas de log do Amazon CloudTrail sempre que a chave do Amazon KMS da instância de banco de dados é acessada.

- DBName: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do banco de dados.

- DBParameterGroups – Uma matriz de objetos DBParameterGroupStatus.

  Fornece a lista de grupo de parâmetros de banco de dados aplicados a essa instância de banco de dados.

- DBSecurityGroups – Uma matriz de objetos DBSecurityGroupMembership.

  Fornece a lista de elementos do grupo de segurança de banco de dados que contém somente os subelementos `DBSecurityGroup.Name` e `DBSecurityGroup.Status`.

- DBSubnetGroup: um objeto DBSubnetGroup.

  Especifica informações sobre o grupo de sub-redes associado à instância de banco de dados, incluindo o nome, a descrição e as sub-redes no grupo de sub-redes.

- DeletionProtection: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica se a instância de banco de dados tem a proteção contra exclusão ativada. A instância não poderá ser excluída quando a proteção contra exclusão estiver ativada. Consulte Excluir uma instância de banco de dados.

- DomainMemberships – Uma matriz de objetos DomainMembership.

  Não suportado

- EnabledCloudwatchLogsExports: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de tipos de logs que essa instância de banco de dados está configurada para exportar para o CloudWatch Logs.

- Endpoint – Um objeto Endpoint.

  Especifica o endpoint de conexão.

- Engine: uma string, do tipo: `string` (uma string codificada em UTF-8).

Fornece o nome do mecanismo de banco de dados a ser usado para essa instância de banco de dados.

- EngineVersion: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Indica a versão do mecanismo do banco de dados.

- EnhancedMonitoringResourceArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) do fluxo de logs do Amazon CloudWatch Logs que recebe os dados de métricas de monitoramento avançado para a instância de banco de dados.

- IAMDatabaseAuthenticationEnabled: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Verdadeiro se a autenticação do Amazon Identity and Access Management (IAM) estiver habilitada. Caso contrário, falso.

- InstanceCreateTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Fornece a data e hora em que a instância de banco de dados foi criada.

- Iops: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica o valor das IOPS (operações de entrada/saída por segundo) provisionadas.

- KmsKeyId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Não compatível: a criptografia para instâncias de banco de dados é gerenciada pelo cluster de banco de dados.

- LatestRestorableTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a data mais recente até a qual um banco de dados pode ser restaurado com restauração pontual.

- LicenseModel: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Informações do modelo de licença dessa instância de banco de dados.

- MonitoringInterval: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O intervalo, em segundos, entre pontos quando as métricas de monitoramento avançado são coletadas para a instância de banco de dados.

- MonitoringRoleArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN da função do IAM que permite que o Neptune envie métricas de monitoramento avançado para o Amazon CloudWatch Logs.

- MultiAZ: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se a instância de banco de dados é uma implantação Multi-AZ.

- PendingModifiedValues: um objeto PendingModifiedValues.

  Especifica que alterações à instância de banco de dados estão pendentes. Esse elemento só é incluído quando as alterações estão pendentes. As alterações específicas são identificadas por subelementos.

- PreferredBackupWindow: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o intervalo de tempo diário durante o qual os backups automatizados serão criados se eles estiverem habilitados, conforme determinado por `BackupRetentionPeriod`.

- PreferredMaintenanceWindow: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o período semanal durante o qual pode ocorrer a manutenção do sistema, em Tempo Universal Coordenado (UTC).

- PromotionTier: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Um valor que especifica a ordem em que uma réplica de leitura é promovida à instância principal após uma falha da instância principal existente.

- PubliclyAccessible: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Este sinalizador não deve mais ser usado.

- ReadReplicaDBClusterIdentifiers: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um ou mais identificadores de clusters de banco de dados que são réplicas de leitura dessa instância de banco de dados.

- ReadReplicaDBInstanceIdentifiers: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um ou mais identificadores das réplicas de leitura associadas a essa instância de banco de dados.

- ReadReplicaSourceDBInstanceIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

Contém o identificador da instância de banco de dados de origem se essa instância de banco de dados for uma réplica de leitura.

- SecondaryAvailabilityZone: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Se houver, especificará o nome da zona de disponibilidade secundária para uma instância de banco de dados com suporte a Multi-AZ
- StatusInfos – Uma matriz de objetos DBInstanceStatusInfo.

  O status da réplica de leitura. Se a instância não for uma réplica de leitura, isso ficará em branco.
- StorageEncrypted: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Não compatível: a criptografia para instâncias de banco de dados é gerenciada pelo cluster de banco de dados.
- StorageType: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o tipo de armazenamento associado à instância de banco de dados.
- TdeCredentialArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN do armazenamento de chaves ao qual a instância está associada para criptografia TDE.
- Timezone: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Sem suporte.
- VpcSecurityGroups – Uma matriz de objetos VpcSecurityGroupMembership.

  Fornece uma lista de elementos do grupo de segurança da VPC à qual a instância de banco de dados pertence.

Erros

- InvalidDBInstanceStateFault
- DBInstanceNotFoundFault

DescribeDBInstances (ação)

O nome da CLI da AWS para essa API é: `describe-db-instances`.

Retorna informações sobre instâncias provisionadas e oferece suporte à paginação.

> **Note**
>
> Essa operação também pode retornar informações para instâncias do Amazon RDS e instâncias do Amazon DocDB.

Solicitação

- DBInstanceIdentifier (na CLI: `--db-instance-identifier`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador da instância fornecido pelo usuário. Se esse parâmetro for especificado, somente as informações da instância de banco de dados específica serão retornadas. Este parâmetro não diferencia maiúsculas de minúsculas.

  Restrições:

  - Se fornecido, deve corresponder ao identificador de uma DBInstance existente.
- Filters (na CLI: `--filters`): uma matriz de objetos Filtro.

  Um filtro que especifica uma ou mais instâncias de banco de dados a serem descritas.

  Filtros suportados:

  - `db-cluster-id` – aceita identificadores de cluster de banco de dados e os Nomes de recursos da Amazon (ARNs) do cluster de banco de dados. A lista de resultados incluirá somente informações sobre as instâncias de banco de dados associadas aos clusters de banco de dados identificados por esses ARNs.
  - `engine` – aceita um nome de mecanismo (como `neptune`) e restringe a lista de resultados a instâncias de banco de dados criadas por esse mecanismo.

  Por exemplo, para invocar essa API na CLI da Amazon e filtrar para que somente instâncias de banco de dados do Neptune sejam exibidas, é possível usar o seguinte comando:

  Example

```
aws neptune describe-db-instances \
            --filters  Name=engine,Values=neptune
```

- Marker (na CLI: `--marker`): uma string, do tipo: `string` (uma string codificada em UTF-8).

Um token de paginação opcional fornecido por uma solicitação `DescribeDBInstances` anterior. Se esse parâmetro for especificado, a resposta incluirá apenas os registros além do marcador, até o valor especificado por `MaxRecords`.

- MaxRecords (na CLI: `--max-records`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número máximo de registros a serem incluídos na resposta. Se existirem mais registros do que o valor `MaxRecords` especificado, um token de paginação chamado de marcador será incluído na resposta para que os resultados restantes possam ser recuperados.

  Padrão: 100

  Restrições: Mínimo 20, máximo 100.

Resposta

- DBInstances – Uma matriz de objetos DBInstance.

  Uma lista de instâncias the section called “DBInstance”.
- Marker: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um token de paginação opcional fornecido por uma solicitação anterior. Se esse parâmetro for especificado, a resposta incluirá apenas os registros além do marcador, até o valor especificado por `MaxRecords`.

Erros

- DBInstanceNotFoundFault

DescribeOrderableDBInstanceOptions (ação)

O nome da CLI da AWS para essa API é: `describe-orderable-db-instance-options`.

Retorna uma lista de opções de instância de banco de dados que podem ser solicitadas para o mecanismo especificado.

Solicitação

- DBInstanceClass (na CLI: `--db-instance-class`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O valor do filtro da classe da instância de banco de dados. Especifique esse parâmetro para mostrar somente as ofertas disponíveis correspondentes à classe da instância de banco de dados especificada.

- Engine (na CLI: `--engine`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do mecanismo para o qual recuperar opções de instância de banco de dados.

- EngineVersion (na CLI: `--engine-version`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O valor do filtro da versão do mecanismo. Especifique esse parâmetro para mostrar somente as ofertas disponíveis correspondentes à versão do mecanismo especificado.

- Filters (na CLI: `--filters`): uma matriz de objetos Filtro.

  Não há suporte para esse parâmetro atualmente.

- LicenseModel (na CLI: `--license-model`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O valor do filtro do modelo de licença. Especifique esse parâmetro para mostrar somente as ofertas disponíveis correspondentes ao modelo de licença especificado.

- Marker (na CLI: `--marker`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um token de paginação opcional fornecido por uma solicitação DescribeOrderableDBInstanceOptions anterior. Se esse parâmetro for especificado, a resposta incluirá apenas os registros além do marcador, até o valor especificado por `MaxRecords`.

- MaxRecords (na CLI: `--max-records`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número máximo de registros a serem incluídos na resposta. Se existirem mais registros do que o valor `MaxRecords` especificado, um token de paginação chamado de marcador será incluído na resposta para que os resultados restantes possam ser recuperados.

  Padrão: 100

  Restrições: Mínimo 20, máximo 100.

- Vpc (na CLI: `--vpc`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  O valor do filtro da VPC. Especifique esse parâmetro para mostrar somente a VPC disponível ou ofertas que não sejam de VPC.

Resposta

- Marker: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um token de paginação opcional fornecido por uma solicitação OrderableDBInstanceOptions anterior. Se esse parâmetro for especificado, a resposta incluirá apenas os registros além do marcador, até o valor especificado por `MaxRecords`.
- OrderableDBInstanceOptions – Uma matriz de objetos OrderableDBInstanceOption.

  Uma estrutura the section called “OrderableDBInstanceOption” que contém informações sobre as opções que podem ser solicitadas para a instância de banco de dados.

DescribeValidDBInstanceModifications (ação)

O nome da CLI da AWS para essa API é: `describe-valid-db-instance-modifications`.

É possível chamar the section called “DescribeValidDBInstanceModifications” para saber quais modificações que você pode fazer em sua instância de banco de dados. Você pode usar essas informações quando chamar the section called “ModifyDBInstance”.

Solicitação

- DBInstanceIdentifier (na CLI: `--db-instance-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador do cliente ou o ARN de sua instância de banco de dados.

Resposta

Informações sobre modificações válidas que você pode fazer à sua instância de banco de dados. Contém o resultado de uma chamada bem-sucedida para a ação the section called

"DescribeValidDBInstanceModifications". Você pode usar essas informações quando chamar the section called "ModifyDBInstance".

- Storage – Uma matriz de objetos ValidStorageOptions.

  Opções de armazenamento válidas para sua instância de banco de dados.

Erros

- DBInstanceNotFoundFault
- InvalidDBInstanceStateFault

Estruturas:

DBInstance (estrutura)

Contém os detalhes de uma instância de banco de dados do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta na ação the section called "DescribeDBInstances".

Campos

- AutoMinorVersionUpgrade: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica que os patches de versões secundárias são aplicados automaticamente.
- AvailabilityZone: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o nome da zona de disponibilidade na qual a instância de banco de dados está localizada.
- BackupRetentionPeriod: é um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica o número de dias durante os quais os DB snapshots automáticos são retidos.
- CACertificateIdentifier: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador do certificado da CA para essa instância DB.
- CopyTagsToSnapshot: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

Especifica se as tags atribuídas são copiadas da instância de banco de dados nos snapshots da instância de banco de dados.

- DBClusterIdentifier: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Se a instância de banco de dados for membro de um cluster de banco de dados, conterá o nome do cluster de banco de dados do qual a instância de banco de dados é membro.
- DBInstanceArn: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) da instância de banco de dados.
- DBInstanceClass: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém o nome da classe de capacidade de computação e memória da instância de banco de dados.
- DBInstanceIdentifier: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um identificador de banco de dados fornecido pelo usuário. Esse identificador é a chave exclusiva que identifica uma instância de banco de dados.
- DbInstancePort: é um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica a porta na qual a instância de banco de dados escuta. Se a instância de banco de dados fizer parte de um cluster de banco de dados, isso poderá ser uma porta diferente da porta do cluster de banco de dados.
- DBInstanceStatus: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o estado atual desse banco de dados.
- DbiResourceId: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador exclusivo e imutável da região da Amazon da instância de banco de dados. Esse identificador é encontrado nas entradas de log do Amazon CloudTrail sempre que a chave do Amazon KMS da instância de banco de dados é acessada.
- DBName: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do banco de dados.
- DBParameterGroups: é uma matriz de objetos DBParameterGroupStatus.

  Fornece a lista de grupo de parâmetros de banco de dados aplicados a essa instância de banco de dados.

- DBSecurityGroups: é uma matriz de objetos DBSecurityGroupMembership.

  Fornece a lista de elementos do grupo de segurança de banco de dados que contém somente os subelementos `DBSecurityGroup.Name` e `DBSecurityGroup.Status`.
- DBSubnetGroup: é um objeto DBSubnetGroup.

  Especifica informações sobre o grupo de sub-redes associado à instância de banco de dados, incluindo o nome, a descrição e as sub-redes no grupo de sub-redes.
- DeletionProtection: é um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica se a instância de banco de dados tem a proteção contra exclusão ativada. A instância não poderá ser excluída quando a proteção contra exclusão estiver ativada. Consulte Excluir uma instância de banco de dados.
- DomainMemberships: é uma matriz de objetos DomainMembership.

  Não suportado
- EnabledCloudwatchLogsExports: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de tipos de logs que essa instância de banco de dados está configurada para exportar para o CloudWatch Logs.
- Endpoint: é um objeto Endpoint.

  Especifica o endpoint de conexão.
- Engine: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o nome do mecanismo de banco de dados a ser usado para essa instância de banco de dados.
- EngineVersion: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Indica a versão do mecanismo do banco de dados.
- EnhancedMonitoringResourceArn: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) do fluxo de logs do Amazon CloudWatch Logs que recebe os dados de métricas de monitoramento avançado para a instância de banco de dados.
- IAMDatabaseAuthenticationEnabled: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

Verdadeiro se a autenticação do Amazon Identity and Access Management (IAM) estiver habilitada. Caso contrário, falso.

- InstanceCreateTime: é um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Fornece a data e hora em que a instância de banco de dados foi criada.

- Iops: é um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica o valor das IOPS (operações de entrada/saída por segundo) provisionadas.

- KmsKeyId: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Não compatível: a criptografia para instâncias de banco de dados é gerenciada pelo cluster de banco de dados.

- LatestRestorableTime: é um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a data mais recente até a qual um banco de dados pode ser restaurado com restauração pontual.

- LicenseModel: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Informações do modelo de licença dessa instância de banco de dados.

- MonitoringInterval: é um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O intervalo, em segundos, entre pontos quando as métricas de monitoramento avançado são coletadas para a instância de banco de dados.

- MonitoringRoleArn: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN da função do IAM que permite que o Neptune envie métricas de monitoramento avançado para o Amazon CloudWatch Logs.

- MultiAZ: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se a instância de banco de dados é uma implantação Multi-AZ.

- PendingModifiedValues: é um objeto PendingModifiedValues.

  Especifica que alterações à instância de banco de dados estão pendentes. Esse elemento só é incluído quando as alterações estão pendentes. As alterações específicas são identificadas por subelementos.

- PreferredBackupWindow: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o intervalo de tempo diário durante o qual os backups automatizados serão criados se eles estiverem habilitados, conforme determinado por `BackupRetentionPeriod`.

- PreferredMaintenanceWindow: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o período semanal durante o qual pode ocorrer a manutenção do sistema, em Tempo Universal Coordenado (UTC).

- PromotionTier: é um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Um valor que especifica a ordem em que uma réplica de leitura é promovida à instância principal após uma falha da instância principal existente.

- PubliclyAccessible: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Este sinalizador não deve mais ser usado.

- ReadReplicaDBClusterIdentifiers: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um ou mais identificadores de clusters de banco de dados que são réplicas de leitura dessa instância de banco de dados.

- ReadReplicaDBInstanceIdentifiers: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um ou mais identificadores das réplicas de leitura associadas a essa instância de banco de dados.

- ReadReplicaSourceDBInstanceIdentifier: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém o identificador da instância de banco de dados de origem se essa instância de banco de dados for uma réplica de leitura.

- SecondaryAvailabilityZone: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Se houver, especificará o nome da zona de disponibilidade secundária para uma instância de banco de dados com suporte a Multi-AZ

- StatusInfos: é uma matriz de objetos DBInstanceStatusInfo.

  O status da réplica de leitura. Se a instância não for uma réplica de leitura, isso ficará em branco.

- StorageEncrypted: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Não compatível: a criptografia para instâncias de banco de dados é gerenciada pelo cluster de banco de dados.
- StorageType: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o tipo de armazenamento associado à instância de banco de dados.
- TdeCredentialArn: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN do armazenamento de chaves ao qual a instância está associada para criptografia TDE.
- Timezone: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Sem suporte.
- VpcSecurityGroups: é uma matriz de objetos VpcSecurityGroupMembership.

  Fornece uma lista de elementos do grupo de segurança da VPC à qual a instância de banco de dados pertence.

`DBInstance` é usado como o elemento de resposta para:

- CreateDBInstance
- DeleteDBInstance
- ModifyDBInstance
- RebootDBInstance

DBInstanceStatusInfo (estrutura)

Fornece uma lista de informações de status para uma instância de banco de dados.

Campos

- Message: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Detalhes do erro se houver um erro para a instância. Se a instância não estiver em um estado de erro, esse valor ficará em branco.
- Normal: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

Valor booliano que será true se a instância estiver funcionando normalmente, ou false se a instância estiver em um estado de erro.

- Status: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Status da instância de banco de dados. Para um StatusType de réplica de leitura, os valores podem ser replicação, erro, interrompida ou encerrada.
- StatusType: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  No momento, esse valor é "read replication".

OrderableDBInstanceOption (estrutura)

Contém uma lista de opções disponíveis para uma instância de banco de dados.

Esse tipo de dados é usado como um elemento de resposta na ação the section called “DescribeOrderableDBInstanceOptions”.

Campos

- AvailabilityZones: é uma matriz de objetos AvailabilityZone.

  Uma lista das zonas de disponibilidade para uma instância de banco de dados.
- DBInstanceClass: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  A classe da instância de banco de dados para uma instância de banco de dados.
- Engine: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo de mecanismo de uma instância de banco de dados.
- EngineVersion: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  A versão do mecanismo de uma instância de banco de dados.
- LicenseModel: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O modelo de licença para uma instância de banco de dados.
- MaxIopsPerDbInstance: é um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Total máximo de IOPS provisionadas para uma instância de banco de dados

- MaxIopsPerGib: é um DoubleOptional, do tipo: `double` (um número de ponto flutuante IEEE 754 de dupla precisão).

  Máximo de IOPS provisionadas por GiB para uma instância de banco de dados
- MaxStorageSize: é um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Tamanho máximo de armazenamento de uma instância de banco de dados.
- MinIopsPerDbInstance: é um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Total mínimo de IOPS provisionadas para uma instância de banco de dados
- MinIopsPerGib: é um DoubleOptional, do tipo: `double` (um número de ponto flutuante IEEE 754 de dupla precisão).

  Mínimo de IOPS provisionadas por GiB para uma instância de banco de dados
- MinStorageSize: é um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Tamanho mínimo de armazenamento de uma instância de banco de dados.
- MultiAZCapable: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica se uma instância de banco de dados tem capacidade para Multi-AZ.
- ReadReplicaCapable: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica se uma instância de banco de dados pode ter uma réplica de leitura.
- StorageType: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Não aplicável. No Neptune, o tipo de armazenamento é gerenciado no nível do cluster de banco de dados.
- SupportsEnhancedMonitoring: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica se uma instância de banco de dados oferece suporte ao monitoramento avançado em intervalos de 1 a 60 segundos.
- SupportsGlobalDatabases: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

Um valor que indica se você pode usar os bancos de dados globais do Neptune com uma combinação específica de outros atributos do mecanismo de banco de dados.

- SupportsIAMDatabaseAuthentication: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica se uma instância de banco de dados oferece suporte à autenticação de banco de dados do IAM.
- SupportsIops: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica se uma instância de banco de dados oferece suporte a IOPS provisionadas.
- SupportsStorageEncryption: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica se uma instância de banco de dados oferece suporte ao armazenamento criptografado.
- Vpc: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica se uma instância de banco de dados está em uma VPC.

PendingModifiedValues (estrutura)

Esse tipo de dados é usado como um elemento de resposta na ação the section called "ModifyDBInstance".

Campos

- AllocatedStorage: é um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Contém o novo tamanho de `AllocatedStorage` para a instância de banco de dados que será aplicado ou está sendo aplicado.
- BackupRetentionPeriod: é um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica o número de dias pendentes durante os quais os backups automatizados são retidos.
- CACertificateIdentifier: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o identificador do certificado da CA para a instância de banco de dados.
- DBInstanceClass: é uma string, do tipo: `string` (uma string codificada em UTF-8).

Contém a nova `DBInstanceClass` para a instância de banco de dados que será aplicada ou está sendo aplicada.

- DBInstanceIdentifier: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém a nova `DBInstanceIdentifier` para a instância de banco de dados que será aplicada ou está sendo aplicada.

- DBSubnetGroupName: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O novo grupo de sub-redes de banco de dados para a instância de banco de dados.

- EngineVersion: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Indica a versão do mecanismo do banco de dados.

- Iops: é um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica o novo valor de IOPS provisionadas para a instância de banco de dados que será aplicado ou está sendo aplicado.

- MultiAZ: é um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica que a instância de banco de dados Single-AZ deve ser alterada apara uma implantação Multi-AZ.

- PendingCloudwatchLogsExports: é um objeto PendingCloudwatchLogsExports.

  Essa estrutura `PendingCloudwatchLogsExports` especifica alterações pendentes em quais logs do CloudWatch estão habilitados e quais estão desabilitados.

- Port: é um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica a porta pendente para a instância de banco de dados.

- StorageType: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Não aplicável. No Neptune, o tipo de armazenamento é gerenciado no nível do cluster de banco de dados.

ValidStorageOptions (estrutura)

Não aplicável. No Neptune, o tipo de armazenamento é gerenciado no nível do cluster de banco de dados.

Campos

- IopsToStorageRatio: é uma matriz de objetos DoubleRange.

  Não aplicável. No Neptune, o tipo de armazenamento é gerenciado no nível do cluster de banco de dados.

- ProvisionedIops: é uma matriz de objetos Intervalo.

  Não aplicável. No Neptune, o tipo de armazenamento é gerenciado no nível do cluster de banco de dados.

- StorageSize: é uma matriz de objetos Intervalo.

  Não aplicável. No Neptune, o tipo de armazenamento é gerenciado no nível do cluster de banco de dados.

- StorageType: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Não aplicável. No Neptune, o tipo de armazenamento é gerenciado no nível do cluster de banco de dados.

ValidDBInstanceModificationsMessage (estrutura)

Informações sobre modificações válidas que você pode fazer à sua instância de banco de dados. Contém o resultado de uma chamada bem-sucedida para a ação the section called "DescribeValidDBInstanceModifications". Você pode usar essas informações quando chamar the section called "ModifyDBInstance".

Campos

- Storage: é uma matriz de objetos ValidStorageOptions.

  Opções de armazenamento válidas para sua instância de banco de dados.

`ValidDBInstanceModificationsMessage` é usado como o elemento de resposta para:

- DescribeValidDBInstanceModifications

API de parâmetros do Neptune

Ações:

- CopyDBParameterGroup (ação)
- CopyDBClusterParameterGroup (ação)
- CreateDBParameterGroup (ação)
- CreateDBClusterParameterGroup (ação)
- DeleteDBParameterGroup (ação)
- DeleteDBClusterParameterGroup (ação)
- ModifyDBParameterGroup (ação)
- ModifyDBClusterParameterGroup (ação)
- ResetDBParameterGroup (ação)
- ResetDBClusterParameterGroup (ação)
- DescribeDBParameters (ação)
- DescribeDBParameterGroups (ação)
- DescribeDBClusterParameters (ação)
- DescribeDBClusterParameterGroups (ação)
- DescribeEngineDefaultParameters (ação)
- DescribeEngineDefaultClusterParameters (ação)

Estruturas:

- Parâmetro (estrutura)
- DBParameterGroup (estrutura)
- DBClusterParameterGroup (estrutura)
- DBParameterGroupStatus (estrutura)

CopyDBParameterGroup (ação)

O nome da CLI da AWS para essa API é: `copy-db-parameter-group`.

Copia o grupo de parâmetros de banco de dados especificado.

Solicitação

- SourceDBParameterGroupIdentifier (na CLI: `--source-db-parameter-group-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador ou o ARN do grupo de parâmetros de banco de dados de origem. Para obter informações sobre a criação de um ARN, consulte Criação de um nome de recurso da Amazon (ARN) do Amazon ARN.

  Restrições:

  - É necessário especificar um grupo de parâmetros de banco de dados válido.
  - É necessário especificar um identificador válido de grupo de parâmetros de banco de dados, por exemplo `my-db-param-group` ou um ARN válido.
- Tags (na CLI: `--tags`): uma matriz de objetos Tag.

  As tags a serem atribuídas ao grupo de parâmetros de banco de dados copiado.
- TargetDBParameterGroupDescription (na CLI: `--target-db-parameter-group-description`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma descrição do grupo de parâmetros de banco de dados copiado.
- TargetDBParameterGroupIdentifier (na CLI: `--target-db-parameter-group-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador do grupo de parâmetros de banco de dados copiado.

  Restrições:

  - Não pode ser nulo, estar vazio ou em branco.
  - Deve conter de 1 a 255 caracteres, incluindo letras, números ou hífens.
  - O primeiro caractere deve ser uma letra.
  - Não podem terminar com um hífen ou conter dois hífens consecutivos.

  Exemplo: `my-db-parameter-group`

Resposta

Contém os detalhes de um grupo de parâmetros de banco de dados do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta na ação the section called "DescribeDBParameterGroups".

- DBParameterGroupArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) do grupo de parâmetros de banco de dados.
- DBParameterGroupFamily: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o nome da família de grupos de parâmetros de banco de dados com a qual esse grupo de parâmetros de banco de dados é compatível.
- DBParameterGroupName: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o nome do grupo de parâmetros de banco de dados.
- Description: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece a descrição especificada pelo cliente para esse grupo de parâmetros de banco de dados.

Erros

- DBParameterGroupNotFoundFault
- DBParameterGroupAlreadyExistsFault
- DBParameterGroupQuotaExceededFault

CopyDBClusterParameterGroup (ação)

O nome da CLI da AWS para essa API é: `copy-db-cluster-parameter-group`.

Copia o grupo de parâmetros de cluster de banco de dados especificado.

Solicitação

- SourceDBClusterParameterGroupIdentifier (na CLI: `--source-db-cluster-parameter-group-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador ou o nome de recurso da Amazon (ARN) do grupo de parâmetros de cluster de banco de dados de origem. Para obter informações sobre a criação de um ARN, consulte Criação de um nome de recurso da Amazon (ARN) do Amazon ARN.

  Restrições:

  - É necessário especificar um grupo de parâmetros de cluster de banco de dados válido.
  - Se o grupo de parâmetros de cluster de banco de dados de origem estiver na mesma região da Amazon que a cópia, especifique um identificador de grupo de parâmetros de banco de dados válido, por exemplo, `my-db-cluster-param-group` ou um ARN válido.
  - Se o grupo de parâmetros de banco de dados de origem estiver em uma região da Amazon diferente da cópia, especifique um ARN do grupo de parâmetros de cluster de banco de dados válido, por exemplo, `arn:aws:rds:us-east-1:123456789012:cluster-pg:custom-cluster-group1`.
- Tags (na CLI: `--tags`): uma matriz de objetos Tag.

  As tags a serem atribuídas ao grupo de parâmetros de cluster de banco de dados copiado.
- TargetDBClusterParameterGroupDescription (na CLI: `--target-db-cluster-parameter-group-description`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma descrição do grupo de parâmetros de cluster de banco de dados copiado.
- TargetDBClusterParameterGroupIdentifier (na CLI: `--target-db-cluster-parameter-group-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador do grupo de parâmetros de cluster de banco de dados copiado.

  Restrições:
  - Não pode ser nulo, estar vazio nem estar em branco
  - Deve conter de 1 a 255 caracteres, incluindo letras, números ou hífens
  - O primeiro caractere deve ser uma letra
  - Não podem terminar com um hífen ou conter dois hífens consecutivos

  Exemplo: `my-cluster-param-group1`

Resposta

Contém os detalhes de um grupo de parâmetros de cluster de banco de dados do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta na ação the section called "DescribeDBClusterParameterGroups".

- DBClusterParameterGroupArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

O nome de recurso da Amazon (ARN) do grupo de parâmetros de cluster de banco de dados.

- DBClusterParameterGroupName: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o nome do parameter group do cluster de banco de dados.

- DBParameterGroupFamily: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o nome da família do parameter group de banco de dados com a qual esse parameter group de cluster de banco de dados é compatível.

- Description: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece a descrição especificada pelo cliente para este parameter group do cluster de banco de dados.

Erros

- DBParameterGroupNotFoundFault
- DBParameterGroupQuotaExceededFault
- DBParameterGroupAlreadyExistsFault

CreateDBParameterGroup (ação)

O nome da CLI da AWS para essa API é: `create-db-parameter-group`.

Cria um novo grupo de parâmetros de banco de dados.

Um grupo de parâmetros de banco de dados é inicialmente criado com os parâmetros padrão para o mecanismo de banco de dados usado pela instância de banco de dados. Para fornecer valores personalizados para qualquer um dos parâmetros, modifique o grupo depois de criá-lo usando ModifyDBParameterGroup. Assim que você criar um grupo de parâmetros de banco de dados, você precisa associá-lo à sua instância de banco de dados usando ModifyDBInstance. Ao associar um novo grupo de parâmetros de banco de dados a uma instância de banco de dados em execução, você precisa reinicializar a instância de banco de dados sem failover para que o novo grupo de parâmetros de banco de dados e as configurações associados entrem em vigor.

> **Important**
>
> Depois de criar um grupo de parâmetros de banco de dados, espere pelo menos 5 minutos para criar sua primeira instância de banco de dados que usa esse grupo de parâmetros como o padrão. Isso permite que o Amazon Neptune complete totalmente a ação de criação antes que o grupo de parâmetros seja usado como padrão para uma nova instância de banco de dados. Isso é especialmente importante para parâmetros que são críticos ao criar o banco de dados padrão para uma instância de banco de dados, como o conjunto de caracteres para o banco de dados padrão definido pelo parâmetro `character_set_database`. Você pode usar a opção Parameter Groups (Grupos de parâmetros) do console do Amazon Neptune ou o comando DescribeDBParameters para verificar se o seu grupo de parâmetros de banco de dados foi criado ou modificado.

Solicitação

- DBParameterGroupFamily (na CLI: `--db-parameter-group-family`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome da família de grupos de parâmetros de banco de dados. Um grupo de parâmetros de banco de dados pode ser associado a uma única família de grupos de parâmetros de banco de dados e pode ser aplicado somente a uma instância de banco de dados que execute um mecanismo de banco de dados e uma versão do mecanismo compatível com essa família de grupos de parâmetros de banco de dados.
- DBParameterGroupName (na CLI: `--db-parameter-group-name`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do grupo de parâmetros de banco de dados.

  Restrições:

  - Deve ter de 1 a 255 letras, números ou hifens.
  - O primeiro caractere deve ser uma letra
  - Não podem terminar com um hífen ou conter dois hífens consecutivos

  > **Note**
  >
  > Esse valor é armazenado como uma string em minúsculas.

- Description (na CLI: `--description`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  A descrição do grupo de parâmetros de banco de dados.
- Tags (na CLI: `--tags`): uma matriz de objetos Tag.

  As tags a serem atribuídas ao novo grupo de parâmetros de banco de dados.

Resposta

Contém os detalhes de um grupo de parâmetros de banco de dados do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta na ação the section called "DescribeDBParameterGroups".

- DBParameterGroupArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) do grupo de parâmetros de banco de dados.
- DBParameterGroupFamily: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o nome da família de grupos de parâmetros de banco de dados com a qual esse grupo de parâmetros de banco de dados é compatível.
- DBParameterGroupName: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o nome do grupo de parâmetros de banco de dados.
- Description: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece a descrição especificada pelo cliente para esse grupo de parâmetros de banco de dados.

Erros

- DBParameterGroupQuotaExceededFault
- DBParameterGroupAlreadyExistsFault

CreateDBClusterParameterGroup (ação)

O nome da CLI da AWS para essa API é: `create-db-cluster-parameter-group`.

Cria um novo grupo de parâmetros de cluster de banco de dados.

Os parâmetros em um grupo de parâmetros de cluster de banco de dados aplicam-se a todas as instâncias em um cluster de banco de dados.

Um grupo de parâmetros de cluster de banco de dados é inicialmente criado com os parâmetros padrão para o mecanismo de banco de dados usado pelas instâncias do cluster de banco de dados. Para fornecer valores personalizados para qualquer um dos parâmetros, modifique o grupo depois de criá-lo usando the section called “ModifyDBClusterParameterGroup”. Assim que você criar um grupo de parâmetros de cluster de banco de dados, você precisa associá-lo ao seu cluster de banco de dados usando the section called “ModifyDBCluster”. Ao associar um novo grupo de parâmetros de cluster de banco de dados a um cluster de banco de dados em execução, você precisa reinicializar as instâncias de banco de dados no cluster de banco de dados sem failover para que o novo grupo de parâmetros de cluster de banco de dados e as configurações associados entrem em vigor.

> **Important**
>
> Depois de criar um grupo de parâmetros de cluster de banco de dados, espere pelo menos 5 minutos para criar seu primeiro cluster de banco de dados usando esse grupo de parâmetros de banco de dados como o padrão. Isso permite que o Amazon Neptune complete totalmente a ação de criação antes que o grupo de parâmetros de cluster de banco de dados seja usado como padrão para um novo cluster de banco de dados. Isso é especialmente importante para parâmetros que são críticos ao criar o banco de dados padrão para um cluster de banco de dados, como o conjunto de caracteres para o banco de dados padrão definido pelo parâmetro `character_set_database`. Você pode usar a opção Parameter Groups (Grupos de parâmetros) do console do Amazon Neptune ou o comando the section called “DescribeDBClusterParameters” para verificar se o seu grupo de parâmetros de cluster de banco de dados foi criado ou modificado.

Solicitação

- DBClusterParameterGroupName (na CLI: `--db-cluster-parameter-group-name`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do grupo de parâmetros de cluster de banco de dados.

  Restrições:

  - Deve ser correspondente ao nome de um DBClusterParameterGroup existente.

> **Note**
>
> Esse valor é armazenado como uma string em minúsculas.

- DBParameterGroupFamily (na CLI: `--db-parameter-group-family`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome da família de grupos de parâmetros de cluster de banco de dados. Um grupo de parâmetros de cluster de banco de dados pode ser associado a uma única família de grupos de parâmetros de cluster de banco de dados e pode ser aplicado somente a um cluster de banco de dados que execute um mecanismo de banco de dados e uma versão do mecanismo compatível com essa família de grupos de parâmetros de cluster de banco de dados.
- Description (na CLI: `--description`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  A descrição do grupo de parâmetros de cluster de banco de dados.
- Tags (na CLI: `--tags`): uma matriz de objetos Tag.

  As tags a serem atribuídas ao novo grupo de parâmetros de cluster de banco de dados.

Resposta

Contém os detalhes de um grupo de parâmetros de cluster de banco de dados do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta na ação the section called "DescribeDBClusterParameterGroups".

- DBClusterParameterGroupArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) do grupo de parâmetros de cluster de banco de dados.
- DBClusterParameterGroupName: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o nome do parameter group do cluster de banco de dados.
- DBParameterGroupFamily: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o nome da família do parameter group de banco de dados com a qual esse parameter group de cluster de banco de dados é compatível.
- Description: uma string, do tipo: `string` (uma string codificada em UTF-8).

Fornece a descrição especificada pelo cliente para este parameter group do cluster de banco de dados.

Erros

- DBParameterGroupQuotaExceededFault
- DBParameterGroupAlreadyExistsFault

DeleteDBParameterGroup (ação)

O nome da CLI da AWS para essa API é: `delete-db-parameter-group`.

Exclui um DBParameterGroup especificado. O DBParameterGroup a ser excluído não pode ser associado a todas as instâncias de banco de dados.

Solicitação

- DBParameterGroupName (na CLI: `--db-parameter-group-name`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do grupo de parâmetros de banco de dados.

  Restrições:

  - Deve ser o nome de um grupo de parâmetros de banco de dados existente.
  - Não é possível excluir um grupo de parâmetros de banco de dados padrão
  - Não pode ser associado a nenhuma instância de banco de dados

Resposta

- Nenhum parâmetro de resposta.

Erros

- InvalidDBParameterGroupStateFault
- DBParameterGroupNotFoundFault

DeleteDBClusterParameterGroup (ação)

O nome da CLI da AWS para essa API é: `delete-db-cluster-parameter-group`.

Exclui um grupo de parâmetros de cluster de banco de dados especificado. O grupo de parâmetros de cluster de banco de dados a ser excluído não pode ser associado a nenhum cluster de banco de dados.

Solicitação

- DBClusterParameterGroupName (na CLI: `--db-cluster-parameter-group-name`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do grupo de parâmetros de cluster de banco de dados.

  Restrições:

  - Deve ser o nome de um grupo de parâmetros de cluster de banco de dados existente.
  - Não é possível excluir um grupo de parâmetros de cluster de banco de dados padrão.
  - Não pode ser associado a nenhum cluster de banco de dados.

Resposta

- Nenhum parâmetro de resposta.

Erros

- InvalidDBParameterGroupStateFault
- DBParameterGroupNotFoundFault

ModifyDBParameterGroup (ação)

O nome da CLI da AWS para essa API é: `modify-db-parameter-group`.

Modifica os parâmetros de um grupo de parâmetros de banco de dados. Para modificar mais de um parâmetro, envie uma lista do seguinte: `ParameterName`, `ParameterValue` e `ApplyMethod`. No máximo, 20 parâmetros pode ser modificado em uma única solicitação.

> **Note**
>
> As alterações em parâmetros dinâmicos são aplicadas imediatamente. As alterações para parâmetros estáticos exigem uma reinicialização sem failover para a instância de banco de dados associada ao grupo de parâmetros para que a alteração entre em vigor.

> **Important**
>
> Depois de modificar um grupo de parâmetros de banco de dados, espere pelo menos 5 minutos para criar sua primeira instância de banco de dados que use esse grupo de parâmetros de banco de dados como o padrão. Isso permite que o Amazon Neptune complete totalmente a ação de modificação antes que o grupo de parâmetros seja usado como padrão para uma nova instância de banco de dados. Isso é especialmente importante para parâmetros que são críticos ao criar o banco de dados padrão para uma instância de banco de dados, como o conjunto de caracteres para o banco de dados padrão definido pelo parâmetro `character_set_database`. Você pode usar a opção Parameter Groups (Grupos de parâmetros) do console do Amazon Neptune ou o comando DescribeDBParameters para verificar se o seu grupo de parâmetros de banco de dados foi criado ou modificado.

Solicitação

- DBParameterGroupName (na CLI: `--db-parameter-group-name`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do grupo de parâmetros de banco de dados.

  Restrições:

  - Se fornecido, deve ser correspondente ao nome de um DBParameterGroup existente.
- Parameters (na CLI: `--parameters`): obrigatório: uma matriz de objetos Parâmetro.

  Uma matriz de nomes de parâmetros, valores e o método de aplicação da atualização de parâmetros. Pelo menos um nome de parâmetro, valor e método de aplicação devem ser fornecidos; argumentos subsequentes são opcionais. No máximo, 20 parâmetros pode ser modificado em uma única solicitação.

Valores válidos (para o método de aplicação): `immediate | pending-reboot`

> **Note**
>
> Você pode usar o valor imediato somente com parâmetros dinâmicos. Você pode usar o valor de reinicialização pendente para parâmetros estáticos e dinâmicos, e as alterações serão aplicadas quando você reiniciar a instância de banco de dados sem failover.

Resposta

- DBParameterGroupName: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o nome do grupo de parâmetros de banco de dados.

Erros

- DBParameterGroupNotFoundFault
- InvalidDBParameterGroupStateFault

ModifyDBClusterParameterGroup (ação)

O nome da CLI da AWS para essa API é: `modify-db-cluster-parameter-group`.

Modifica os parâmetros de um grupo de parâmetros de cluster de banco de dados. Para modificar mais de um parâmetro, envie uma lista do seguinte: `ParameterName`, `ParameterValue` e `ApplyMethod`. No máximo, 20 parâmetros pode ser modificado em uma única solicitação.

> **Note**
>
> As alterações em parâmetros dinâmicos são aplicadas imediatamente. As alterações em parâmetros estáticos exigem uma reinicialização sem failover para o cluster de banco de dados associado ao grupo de parâmetros para que a alteração entre em vigor.

> **Important**
>
> Depois de criar um grupo de parâmetros de cluster de banco de dados, espere pelo menos 5 minutos para criar seu primeiro cluster de banco de dados usando esse grupo de parâmetros de banco de dados como o padrão. Isso permite que o Amazon Neptune complete totalmente a ação de criação antes que o grupo de parâmetros seja usado como padrão para um novo cluster de banco de dados. Isso é especialmente importante para parâmetros que são críticos ao criar o banco de dados padrão para um cluster de banco de dados, como o conjunto de caracteres para o banco de dados padrão definido pelo parâmetro `character_set_database`. Você pode usar a opção Parameter Groups (Grupos de parâmetros) do console do Amazon Neptune ou o comando the section called "DescribeDBClusterParameters" para verificar se o seu grupo de parâmetros de cluster de banco de dados foi criado ou modificado.

Solicitação

- DBClusterParameterGroupName (na CLI: `--db-cluster-parameter-group-name`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do grupo de parâmetros de cluster de banco de dados a ser modificado.
- Parameters (na CLI: `--parameters`): obrigatório: uma matriz de objetos Parâmetro.

  Uma lista de parâmetros no grupo de parâmetros de cluster de banco de dados a ser modificado.

Resposta

- DBClusterParameterGroupName: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do grupo de parâmetros de cluster de banco de dados.

  Restrições:

  - Deve ter de 1 a 255 letras ou números.
  - O primeiro caractere deve ser uma letra
  - Não podem terminar com um hífen ou conter dois hífens consecutivos

> **Note**
>
> Esse valor é armazenado como uma string em minúsculas.

Erros

- DBParameterGroupNotFoundFault
- InvalidDBParameterGroupStateFault

ResetDBParameterGroup (ação)

O nome da CLI da AWS para essa API é: `reset-db-parameter-group`.

Modifica os parâmetros de um grupo de parâmetros de banco de dados para o valor padrão do mecanismo/sistema. Para redefinir parâmetros específicos, forneça uma lista do seguinte: `ParameterName` e `ApplyMethod`. Para redefinir todo o grupo de parâmetros de banco de dados, especifique o nome `DBParameterGroup` e os parâmetros `ResetAllParameters`. Quando você redefine todo o grupo, os parâmetros dinâmicos são atualizados imediatamente, e os parâmetros estáticos são definidos como `pending-reboot` para entrar em vigor na próxima reinicialização da instância de banco de dados ou na solicitação `RebootDBInstance`.

Solicitação

- DBParameterGroupName (na CLI: `--db-parameter-group-name`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do grupo de parâmetros de banco de dados.

  Restrições:

  - Deve ser correspondente ao nome de um DBParameterGroup existente.
- Parameters (na CLI: `--parameters`): uma matriz de objetos Parâmetro.

  Para redefinir todo o grupo de parâmetros de banco de dados, especifique o nome `DBParameterGroup` e os parâmetros `ResetAllParameters`. Para redefinir parâmetros específicos, forneça uma lista do seguinte: `ParameterName` e `ApplyMethod`. No máximo, 20 parâmetros pode ser modificado em uma única solicitação.

Valores válidos (para o método de aplicação): `pending-reboot`

- ResetAllParameters (na CLI: `--reset-all-parameters`): um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se é necessário redefinir todos os parâmetros do grupo de parâmetros de banco de dados como valores padrão (`true`) ou não (`false`).

  Padrão: `true`

Resposta

- DBParameterGroupName: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o nome do grupo de parâmetros de banco de dados.

Erros

- InvalidDBParameterGroupStateFault
- DBParameterGroupNotFoundFault

ResetDBClusterParameterGroup (ação)

O nome da CLI da AWS para essa API é: `reset-db-cluster-parameter-group`.

Modifica os parâmetros de um grupo de parâmetros de cluster de banco de dados para o valor padrão. Para redefinir parâmetros específicos, envie uma lista do seguinte: `ParameterName` e `ApplyMethod`. Para redefinir todo o grupo de parâmetros de cluster de banco de dados, especifique o `DBClusterParameterGroupName` e os parâmetros `ResetAllParameters`.

Quando você redefine todo o grupo, os parâmetros dinâmicos são atualizados imediatamente, e os parâmetros estáticos são definidos como `pending-reboot` para entrar em vigor na próxima reinicialização da instância de banco de dados ou na solicitação the section called “RebootDBInstance”. Você deve chamar the section called “RebootDBInstance” para cada instância de banco de dados no cluster de banco de dados à qual deseja aplicar o parâmetro estático atualizado.

Solicitação

- DBClusterParameterGroupName (na CLI: `--db-cluster-parameter-group-name`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do grupo de parâmetros de cluster de banco de dados a ser redefinido.

- Parameters (na CLI: `--parameters`): uma matriz de objetos Parâmetro.

  Uma lista de nomes de parâmetros no grupo de parâmetros de cluster de banco de dados a serem redefinidos como valores padrão. Você não poderá usar esse parâmetro se o parâmetro `ResetAllParameters` estiver definido como `true`.

- ResetAllParameters (na CLI: `--reset-all-parameters`): um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Um valor definido como `true` para redefinir todos os parâmetros do grupo de parâmetros de cluster de banco de dados como seus valores padrão, e `false` caso contrário. Você não poderá usar esse parâmetro se houver uma lista de nomes de parâmetros especificados para o parâmetro `Parameters`.

Resposta

- DBClusterParameterGroupName: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do grupo de parâmetros de cluster de banco de dados.

  Restrições:

  - Deve ter de 1 a 255 letras ou números.
  - O primeiro caractere deve ser uma letra
  - Não podem terminar com um hífen ou conter dois hífens consecutivos

  > **Note**
  >
  > Esse valor é armazenado como uma string em minúsculas.

Erros

- InvalidDBParameterGroupStateFault
- DBParameterGroupNotFoundFault

DescribeDBParameters (ação)

O nome da CLI da AWS para essa API é: `describe-db-parameters`.

Retorna a lista de parâmetros detalhada de um grupo de parâmetros de banco de dados.

Solicitação

- DBParameterGroupName (na CLI: `--db-parameter-group-name`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de um grupo de parâmetros de banco de dados específico do qual retornar detalhes.

  Restrições:

  - Se fornecido, deve ser correspondente ao nome de um DBParameterGroup existente.
- Filters (na CLI: `--filters`): uma matriz de objetos Filtro.

  Não há suporte atualmente para esse parâmetro.
- Marker (na CLI: `--marker`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um token de paginação opcional fornecido por uma solicitação `DescribeDBParameters` anterior. Se esse parâmetro for especificado, a resposta incluirá apenas os registros além do marcador, até o valor especificado por `MaxRecords`.
- MaxRecords (na CLI: `--max-records`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número máximo de registros a serem incluídos na resposta. Se existirem mais registros do que o valor `MaxRecords` especificado, um token de paginação chamado de marcador será incluído na resposta para que os resultados restantes possam ser recuperados.

  Padrão: 100

  Restrições: mínimo 20, máximo 100.
- Source (na CLI: `--source`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Os tipos de parâmetros a serem retornados.

  Padrão: todos os tipos de parâmetros são retornados

  Valores válidos: `user | system | engine-default`

Resposta

- Marker: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um token de paginação opcional fornecido por uma solicitação anterior. Se esse parâmetro for especificado, a resposta incluirá apenas os registros além do marcador, até o valor especificado por `MaxRecords`.
- Parameters – Uma matriz de objetos Parâmetro.

  Uma lista de valores the section called “Parâmetro”.

Erros

- DBParameterGroupNotFoundFault

DescribeDBParameterGroups (ação)

O nome da CLI da AWS para essa API é: `describe-db-parameter-groups`.

Retorna uma lista de descrições de `DBParameterGroup`. Se um `DBParameterGroupName` for especificado, a lista conterá apenas a descrição do grupo de parâmetros de banco de dados especificado.

Solicitação

- DBParameterGroupName (na CLI: `--db-parameter-group-name`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de um grupo de parâmetros de banco de dados específico do qual retornar detalhes.

  Restrições:

  - Se for fornecido, deverá corresponder ao nome de um DBClusterParameterGroup existente.
- Filters (na CLI: `--filters`): uma matriz de objetos Filtro.

  Não há suporte atualmente para esse parâmetro.
- Marker (na CLI: `--marker`): uma string, do tipo: `string` (uma string codificada em UTF-8).

Um token de paginação opcional fornecido por uma solicitação `DescribeDBParameterGroups` anterior. Se esse parâmetro for especificado, a resposta incluirá apenas os registros além do marcador, até o valor especificado por `MaxRecords`.

- MaxRecords (na CLI: `--max-records`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número máximo de registros a serem incluídos na resposta. Se existirem mais registros do que o valor `MaxRecords` especificado, um token de paginação chamado de marcador será incluído na resposta para que os resultados restantes possam ser recuperados.

  Padrão: 100

  Restrições: mínimo 20, máximo 100.

Resposta

- DBParameterGroups – Uma matriz de objetos DBParameterGroup.

  Uma lista de instâncias the section called “DBParameterGroup”.
- Marker: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um token de paginação opcional fornecido por uma solicitação anterior. Se esse parâmetro for especificado, a resposta incluirá apenas os registros além do marcador, até o valor especificado por `MaxRecords`.

Erros

- DBParameterGroupNotFoundFault

DescribeDBClusterParameters (ação)

O nome da CLI da AWS para essa API é: `describe-db-cluster-parameters`.

Retorna a lista de parâmetros detalhada de um grupo de parâmetros de cluster de banco de dados.

Solicitação

- DBClusterParameterGroupName (na CLI: `--db-cluster-parameter-group-name`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de um grupo de parâmetros de cluster de banco de dados específico do qual retornar detalhes de parâmetros.

  Restrições:

  - Se for fornecido, deverá corresponder ao nome de um DBClusterParameterGroup existente.
- Filters (na CLI: `--filters`): uma matriz de objetos Filtro.

  Não há suporte atualmente para esse parâmetro.
- Marker (na CLI: `--marker`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um token de paginação opcional fornecido por uma solicitação `DescribeDBClusterParameters` anterior. Se esse parâmetro for especificado, a resposta incluirá apenas os registros além do marcador, até o valor especificado por `MaxRecords`.
- MaxRecords (na CLI: `--max-records`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número máximo de registros a serem incluídos na resposta. Se existirem mais registros do que o valor `MaxRecords` especificado, um token de paginação chamado de marcador será incluído na resposta para que os resultados restantes possam ser recuperados.

  Padrão: 100

  Restrições: mínimo 20, máximo 100.
- Source (na CLI: `--source`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um valor que indica para retornar apenas parâmetros para uma origem específica. As origens de parâmetros podem ser `engine`, `service` ou `customer`.

Resposta

- Marker: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um token de paginação opcional fornecido por uma solicitação DescribeDBClusterParameters anterior. Se esse parâmetro for especificado, a resposta incluirá apenas os registros além do marcador, até o valor especificado por `MaxRecords`.

- Parameters – Uma matriz de objetos Parâmetro.

  Fornece uma lista de parâmetros para o grupo de parâmetros de cluster de banco de dados.

Erros

- DBParameterGroupNotFoundFault

DescribeDBClusterParameterGroups (ação)

O nome da CLI da AWS para essa API é: `describe-db-cluster-parameter-groups`.

Retorna uma lista de descrições de `DBClusterParameterGroup`. Se um parâmetro `DBClusterParameterGroupName` for especificado, a lista conterá apenas a descrição do grupo de parâmetros de cluster de banco de dados especificado.

Solicitação

- DBClusterParameterGroupName (na CLI: `--db-cluster-parameter-group-name`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de um grupo de parâmetros de cluster de banco de dados específico do qual retornar detalhes.

  Restrições:

  - Se for fornecido, deverá corresponder ao nome de um DBClusterParameterGroup existente.
- Filters (na CLI: `--filters`): uma matriz de objetos Filtro.

  Não há suporte atualmente para esse parâmetro.
- Marker (na CLI: `--marker`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um token de paginação opcional fornecido por uma solicitação `DescribeDBClusterParameterGroups` anterior. Se esse parâmetro for especificado, a resposta incluirá apenas os registros além do marcador, até o valor especificado por `MaxRecords`.
- MaxRecords (na CLI: `--max-records`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

O número máximo de registros a serem incluídos na resposta. Se existirem mais registros do que o valor `MaxRecords` especificado, um token de paginação chamado de marcador será incluído na resposta para que os resultados restantes possam ser recuperados.

Padrão: 100

Restrições: mínimo 20, máximo 100.

Resposta

- DBClusterParameterGroups – Uma matriz de objetos DBClusterParameterGroup.

  Uma lista de grupos de parâmetros de cluster de banco de dados.
- Marker: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um token de paginação opcional fornecido por uma solicitação `DescribeDBClusterParameterGroups` anterior. Se esse parâmetro for especificado, a resposta incluirá apenas os registros além do marcador, até o valor especificado por `MaxRecords`.

Erros

- DBParameterGroupNotFoundFault

DescribeEngineDefaultParameters (ação)

O nome da CLI da AWS para essa API é: `describe-engine-default-parameters`.

Retorna as informações sobre o mecanismo padrão e parâmetros do sistema do mecanismo de banco de dados especificado.

Solicitação

- DBParameterGroupFamily (na CLI: `--db-parameter-group-family`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome da família de grupos de parâmetros de banco de dados.
- Filters (na CLI: `--filters`): uma matriz de objetos Filtro.

Sem suporte no momento.

- Marker (na CLI: `--marker`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um token de paginação opcional fornecido por uma solicitação `DescribeEngineDefaultParameters` anterior. Se esse parâmetro for especificado, a resposta incluirá apenas os registros além do marcador, até o valor especificado por `MaxRecords`.

- MaxRecords (na CLI: `--max-records`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número máximo de registros a serem incluídos na resposta. Se existirem mais registros do que o valor `MaxRecords` especificado, um token de paginação chamado de marcador será incluído na resposta para que os resultados restantes possam ser recuperados.

  Padrão: 100

  Restrições: mínimo 20, máximo 100.

Resposta

Contém o resultado de uma invocação bem-sucedida da ação the section called "DescribeEngineDefaultParameters".

- DBParameterGroupFamily: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o nome da família de grupos de parâmetros de banco de dados à qual os parâmetros padrão do mecanismo se aplicam.

- Marker: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um token de paginação opcional fornecido por uma solicitação EngineDefaults anterior. Se esse parâmetro for especificado, a resposta incluirá apenas os registros além do marcador, até o valor especificado por `MaxRecords`.

- Parameters – Uma matriz de objetos Parâmetro.

  Contém uma lista de parâmetros padrão do mecanismo.

DescribeEngineDefaultClusterParameters (ação)

O nome da CLI da AWS para essa API é: `describe-engine-default-cluster-parameters`.

Retorna as informações sobre o mecanismo padrão e os parâmetros do sistema do mecanismo de banco de dados do cluster.

Solicitação

- DBParameterGroupFamily (na CLI: `--db-parameter-group-family`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome da família de grupos de parâmetros de cluster de banco de dados da qual retornar informações de parâmetros do mecanismo.
- Filters (na CLI: `--filters`): uma matriz de objetos Filtro.

  Não há suporte atualmente para esse parâmetro.
- Marker (na CLI: `--marker`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um token de paginação opcional fornecido por uma solicitação `DescribeEngineDefaultClusterParameters` anterior. Se esse parâmetro for especificado, a resposta incluirá apenas os registros além do marcador, até o valor especificado por `MaxRecords`.
- MaxRecords (na CLI: `--max-records`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número máximo de registros a serem incluídos na resposta. Se existirem mais registros do que o valor `MaxRecords` especificado, um token de paginação chamado de marcador será incluído na resposta para que os resultados restantes possam ser recuperados.

  Padrão: 100

  Restrições: mínimo 20, máximo 100.

Resposta

Contém o resultado de uma invocação bem-sucedida da ação the section called “DescribeEngineDefaultParameters”.

- DBParameterGroupFamily: uma string, do tipo: `string` (uma string codificada em UTF-8).

Especifica o nome da família de grupos de parâmetros de banco de dados à qual os parâmetros padrão do mecanismo se aplicam.

- Marker: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um token de paginação opcional fornecido por uma solicitação EngineDefaults anterior. Se esse parâmetro for especificado, a resposta incluirá apenas os registros além do marcador, até o valor especificado por `MaxRecords`.
- Parameters – Uma matriz de objetos Parâmetro.

  Contém uma lista de parâmetros padrão do mecanismo.

Estruturas:

Parâmetro (estrutura)

Especifica um parâmetro.

Campos

- AllowedValues: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o intervalo válido de valores para o parâmetro.
- ApplyMethod: é um ApplyMethod, do tipo `string`: (uma string codificada em UTF-8).

  Indica quando aplicar atualizações de parâmetros.
- ApplyType: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o tipo de parâmetros específicos do mecanismo.
- DataType: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o tipo de dados válidos para o parâmetro.
- Description: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece uma descrição do parâmetro.
- IsModifiable: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica se o parâmetro pode (`true`) ou não (`false`) ser modificado. Alguns parâmetros têm implicações de segurança ou operacionais que os impedem de ser alterados.

- MinimumEngineVersion: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  A versão mais antiga do mecanismo à qual o parâmetro pode ser aplicado.
- ParameterName: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o nome do parâmetro.
- ParameterValue: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o valor do parâmetro.
- Source: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Indica a origem do valor do parâmetro.

DBParameterGroup (estrutura)

Contém os detalhes de um grupo de parâmetros de banco de dados do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta na ação the section called "DescribeDBParameterGroups".

Campos

- DBParameterGroupArn: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) do grupo de parâmetros de banco de dados.
- DBParameterGroupFamily: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o nome da família de grupos de parâmetros de banco de dados com a qual esse grupo de parâmetros de banco de dados é compatível.
- DBParameterGroupName: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o nome do grupo de parâmetros de banco de dados.
- Description: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece a descrição especificada pelo cliente para esse grupo de parâmetros de banco de dados.

`DBParameterGroup` é usado como o elemento de resposta para:

- CopyDBParameterGroup

- CreateDBParameterGroup

DBClusterParameterGroup (estrutura)

Contém os detalhes de um grupo de parâmetros de cluster de banco de dados do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta na ação the section called "DescribeDBClusterParameterGroups".

Campos

- DBClusterParameterGroupArn: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) do grupo de parâmetros de cluster de banco de dados.
- DBClusterParameterGroupName: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o nome do parameter group do cluster de banco de dados.
- DBParameterGroupFamily: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o nome da família do parameter group de banco de dados com a qual esse parameter group de cluster de banco de dados é compatível.
- Description: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece a descrição especificada pelo cliente para este parameter group do cluster de banco de dados.

`DBClusterParameterGroup` é usado como o elemento de resposta para:

- CopyDBClusterParameterGroup
- CreateDBClusterParameterGroup

DBParameterGroupStatus (estrutura)

O status do grupo de parâmetros de banco de dados.

Esse tipo de dados é usado como um elemento de resposta nas seguintes ações:

- the section called "CreateDBInstance"

- the section called “DeleteDBInstance”
- the section called “ModifyDBInstance”
- the section called “RebootDBInstance”

Campos

- DBParameterGroupName: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do grupo de parâmetros de banco de dados.
- ParameterApplyStatus: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O status de atualizações de parâmetros.

API da sub-rede do Neptune

Ações:

- CreateDBSubnetGroup (ação)
- DeleteDBSubnetGroup (ação)
- ModifyDBSubnetGroup (ação)
- DescribeDBSubnetGroups (ação)

Estruturas:

- Sub-rede (estrutura)
- DBSubnetGroup (estrutura)

CreateDBSubnetGroup (ação)

> O nome da CLI da AWS para essa API é: `create-db-subnet-group`.

Cria um novo grupo de sub-redes de banco de dados. Os grupos de sub-redes de banco de dados devem conter pelo menos uma sub-rede em pelo menos duas zonas de disponibilidade na região da Amazon.

Solicitação

- DBSubnetGroupDescription (na CLI: `--db-subnet-group-description`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  A descrição do grupo de sub-rede de banco de dados.
- DBSubnetGroupName (na CLI: `--db-subnet-group-name`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do grupo de sub-rede de banco de dados. Esse valor é armazenado como uma string em minúsculas.

  Restrições: deve conter não mais do que 255 letras, números, pontos, sublinhados, espaços ou hífens. Ele não deve ser padrão.

  Exemplo: `mySubnetgroup`
- SubnetIds (na CLI: `--subnet-ids`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Os IDs de sub-rede do EC2 do grupo de sub-redes de banco de dados.
- Tags (na CLI: `--tags`): uma matriz de objetos Tag.

  As tags a serem atribuídas ao novo grupo de sub-redes de banco de dados.

Resposta

Contém os detalhes de um grupo de sub-redes de banco de dados do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta na ação the section called "DescribeDBSubnetGroups".

- DBSubnetGroupArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) do grupo de sub-redes de banco de dados.
- DBSubnetGroupDescription: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece a descrição do grupo de sub-redes de banco de dados.
- DBSubnetGroupName: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do grupo de sub-redes de banco de dados.
- SubnetGroupStatus: uma string, do tipo: `string` (uma string codificada em UTF-8).

Fornece o status do grupo de sub-redes de banco de dados.

- Subnets – Uma matriz de objetos Sub-rede.

  Contém uma lista de elementos the section called “Sub-rede”.

- VpcId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o VpcId do grupo de sub-redes de banco de dados.

Erros

- DBSubnetGroupAlreadyExistsFault
- DBSubnetGroupQuotaExceededFault
- DBSubnetQuotaExceededFault
- DBSubnetGroupDoesNotCoverEnoughAZs
- InvalidSubnet

DeleteDBSubnetGroup (ação)

O nome da CLI da AWS para essa API é: `delete-db-subnet-group`.

Exclui um grupo de sub-redes de banco de dados.

> **Note**
>
> O grupo de sub-redes de banco de dados especificado não deve ser associado a nenhuma instância de banco de dados.

Solicitação

- DBSubnetGroupName (na CLI: `--db-subnet-group-name`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do grupo de sub-redes de banco de dados a ser excluído.

> **Note**
>
> Você não pode excluir o grupo de sub-redes padrão.

Restrições:

Restrições: deve corresponder ao nome de um DBSubnetGroup existente. Ele não deve ser padrão.

Exemplo: `mySubnetgroup`

Resposta

- Nenhum parâmetro de resposta.

Erros

- InvalidDBSubnetGroupStateFault
- InvalidDBSubnetStateFault
- DBSubnetGroupNotFoundFault

ModifyDBSubnetGroup (ação)

O nome da CLI da AWS para essa API é: `modify-db-subnet-group`.

Modifica um grupo de sub-redes de banco de dados existente. Os grupos de sub-redes de banco de dados devem conter pelo menos uma sub-rede em pelo menos duas zonas de disponibilidade na região da Amazon.

Solicitação

- DBSubnetGroupDescription (na CLI: `--db-subnet-group-description`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  A descrição do grupo de sub-rede de banco de dados.

- DBSubnetGroupName (na CLI: `--db-subnet-group-name`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do grupo de sub-rede de banco de dados. Esse valor é armazenado como uma string em minúsculas. Você não pode modificar o grupo de sub-redes padrão.

  Restrições: deve corresponder ao nome de um DBSubnetGroup existente. Ele não deve ser padrão.

  Exemplo: `mySubnetgroup`

- SubnetIds (na CLI: `--subnet-ids`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Os IDs de sub-rede do EC2 do grupo de sub-redes de banco de dados.

Resposta

Contém os detalhes de um grupo de sub-redes de banco de dados do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta na ação the section called "DescribeDBSubnetGroups".

- DBSubnetGroupArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) do grupo de sub-redes de banco de dados.

- DBSubnetGroupDescription: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece a descrição do grupo de sub-redes de banco de dados.

- DBSubnetGroupName: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do grupo de sub-redes de banco de dados.

- SubnetGroupStatus: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o status do grupo de sub-redes de banco de dados.

- Subnets – Uma matriz de objetos Sub-rede.

  Contém uma lista de elementos the section called "Sub-rede".

- VpcId: uma string, do tipo: `string` (uma string codificada em UTF-8).

Fornece o VpcId do grupo de sub-redes de banco de dados.

Erros

- DBSubnetGroupNotFoundFault
- DBSubnetQuotaExceededFault
- SubnetAlreadyInUse
- DBSubnetGroupDoesNotCoverEnoughAZs
- InvalidSubnet

DescribeDBSubnetGroups (ação)

O nome da CLI da AWS para essa API é: `describe-db-subnet-groups`.

Retorna uma lista de descrições de DBSubnetGroup. Se um DBSubnetGroupName for especificado, a lista conterá apenas as descrições do DBSubnetGroup especificado.

Para obter uma visão geral de intervalos CIDR, vá até o Wikipedia Tutorial.

Solicitação

- DBSubnetGroupName (na CLI: `--db-subnet-group-name`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do grupo de sub-redes de banco de dados do qual retornar detalhes.
- Filters (na CLI: `--filters`): uma matriz de objetos Filtro.

  Não há suporte atualmente para esse parâmetro.
- Marker (na CLI: `--marker`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um token de paginação opcional fornecido por uma solicitação DescribeDBSubnetGroups anterior. Se esse parâmetro for especificado, a resposta incluirá apenas os registros além do marcador, até o valor especificado por `MaxRecords`.
- MaxRecords (na CLI: `--max-records`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

O número máximo de registros a serem incluídos na resposta. Se existirem mais registros do que o valor `MaxRecords` especificado, um token de paginação chamado de marcador será incluído na resposta para que os resultados restantes possam ser recuperados.

Padrão: 100

Restrições: mínimo 20, máximo 100.

Resposta

- DBSubnetGroups – Uma matriz de objetos DBSubnetGroup.

  Uma lista de instâncias the section called “DBSubnetGroup”.
- Marker: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um token de paginação opcional fornecido por uma solicitação anterior. Se esse parâmetro for especificado, a resposta incluirá apenas os registros além do marcador, até o valor especificado por `MaxRecords`.

Erros

- DBSubnetGroupNotFoundFault

Estruturas:

Sub-rede (estrutura)

Especifica uma sub-rede.

Esse tipo de dados é usado como um elemento de resposta na ação the section called “DescribeDBSubnetGroups”.

Campos

- SubnetAvailabilityZone: é um objeto AvailabilityZone.

  Especifica a zona de disponibilidade do EC2 na qual se encontra a sub-rede.
- SubnetIdentifier: é uma string, do tipo: `string` (uma string codificada em UTF-8).

Especifica o identificador da sub-rede.

- SubnetStatus: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o status da sub-rede.

DBSubnetGroup (estrutura)

Contém os detalhes de um grupo de sub-redes de banco de dados do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta na ação the section called "DescribeDBSubnetGroups".

Campos

- DBSubnetGroupArn: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) do grupo de sub-redes de banco de dados.
- DBSubnetGroupDescription: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece a descrição do grupo de sub-redes de banco de dados.
- DBSubnetGroupName: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do grupo de sub-redes de banco de dados.
- SubnetGroupStatus: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o status do grupo de sub-redes de banco de dados.
- Subnets: é uma matriz de objetos Sub-rede.

  Contém uma lista de elementos the section called "Sub-rede".
- VpcId: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o VpcId do grupo de sub-redes de banco de dados.

`DBSubnetGroup` é usado como o elemento de resposta para:

- CreateDBSubnetGroup
- ModifyDBSubnetGroup

API de snapshots do Neptune

Ações:

- CreateDBClusterSnapshot (ação)
- DeleteDBClusterSnapshot (ação)
- CopyDBClusterSnapshot (ação)
- ModifyDBClusterSnapshotAttribute (ação)
- RestoreDBClusterFromSnapshot (ação)
- RestoreDBClusterToPointInTime (ação)
- DescribeDBClusterSnapshots (ação)
- DescribeDBClusterSnapshotAttributes (ação)

Estruturas:

- DBClusterSnapshot (estrutura)
- DBClusterSnapshotAttribute (estrutura)
- DBClusterSnapshotAttributesResult (estrutura)

CreateDBClusterSnapshot (ação)

> O nome da CLI da AWS para essa API é: `create-db-cluster-snapshot`.

Cria um snapshot de um cluster de banco de dados.

Solicitação

- DBClusterIdentifier (na CLI: `--db-cluster-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador do cluster de banco de dados para o qual criar um snapshot. Esse parâmetro não diferencia maiúsculas de minúsculas.

  Restrições:

  - Deve ser o identificador de um DBCluster existente.

  Exemplo: `my-cluster1`

- DBClusterSnapshotIdentifier (na CLI: `--db-cluster-snapshot-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador do snapshot do cluster de banco de dados. Este parâmetro é armazenado como uma string com letras minúsculas.

  Restrições:

  - Deve conter de 1 a 63 caracteres, incluindo letras, números ou hífens.
  - O primeiro caractere deve ser uma letra.
  - Não podem terminar com um hífen ou conter dois hífens consecutivos.

  Exemplo: `my-cluster1-snapshot1`
- Tags (na CLI: `--tags`): uma matriz de objetos Tag.

  As tags a serem atribuídas ao snapshot do cluster de banco de dados.

Resposta

Contém os detalhes de um snapshot de cluster de banco de dados do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta na ação the section called "DescribeDBClusterSnapshots".

- AllocatedStorage: um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica o tamanho de armazenamento alocado em gibibytes (GiB).
- AvailabilityZones: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece a lista de zonas de disponibilidade do EC2 nas quais as instâncias no snapshot de cluster de banco de dados podem ser restauradas.
- ClusterCreateTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a hora em que o cluster de banco de dados foi criado, no Tempo Universal Coordenado (UTC).
- DBClusterIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o identificador do cluster de banco de dados do qual esse snapshot de cluster de banco de dados foi criado.

- DBClusterSnapshotArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) do snapshot do cluster de banco de dados.

- DBClusterSnapshotIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o identificador de um snapshot de cluster de banco de dados. Deve corresponder ao identificador de um snapshot existente.

  Depois de restaurar um cluster de banco de dados usando um `DBClusterSnapshotIdentifier`, é necessário especificar o mesmo `DBClusterSnapshotIdentifier` para qualquer atualização futura para o cluster de banco de dados. Quando você especifica essa propriedade para uma atualização, o cluster de banco de dados não é restaurado do snapshot novamente, e os dados no banco de dados não são alterados.

  No entanto, se você não especificar o `DBClusterSnapshotIdentifier`, será criado um cluster de banco de dados vazio e o cluster de banco de dados original será excluído. Se você especificar uma propriedade diferente da propriedade de restauração de snapshot anterior, o cluster de banco de dados será restaurado do snapshot especificado pelo `DBClusterSnapshotIdentifier` e o cluster de banco de dados original será excluído.

- Engine: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o nome do mecanismo de banco de dados.

- EngineVersion: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece a versão do mecanismo de banco de dados para esse snapshot do cluster de banco de dados.

- IAMDatabaseAuthenticationEnabled: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  True se o mapeamento de contas do Amazon Identity and Access Management (IAM) para contas de banco de dados estiver habilitado. Caso contrário, false.

- KmsKeyId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Se `StorageEncrypted` for verdadeiro, o identificador da chave do Amazon KMS para o cluster de banco de dados criptografado.

- LicenseModel: uma string, do tipo: `string` (uma string codificada em UTF-8).

Fornece as informações do modelo de licença para esse snapshot do cluster de banco de dados.

- PercentProgress: um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica a porcentagem dos dados estimados transferidos.

- Port: um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica a porta na qual o cluster de banco de dados estava realizando a escuta no momento do snapshot.

- SnapshotCreateTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Fornece a hora em que o snapshot foi criado, no Tempo Universal Coordenado (UTC).

- SnapshotType: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o tipo do snapshot do cluster de banco de dados.

- SourceDBClusterSnapshotArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Se o snapshot do cluster de banco de dados foi copiado de um snapshot do cluster de banco de dados de origem, o nome de recurso da Amazon (ARN) do snapshot do cluster de banco de dados de origem, caso contrário, um valor nulo.

- Status: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o status desse snapshot de cluster de banco de dados.

- StorageEncrypted: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se o snapshot do cluster de banco de dados está criptografado.

- StorageType: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo de armazenamento associado ao snapshot de cluster de banco de dados.

- VpcId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o ID da VPC associada ao snapshot de cluster de banco de dados.

Erros

- DBClusterSnapshotAlreadyExistsFault
- InvalidDBClusterStateFault

- DBClusterNotFoundFault
- SnapshotQuotaExceededFault
- InvalidDBClusterSnapshotStateFault

DeleteDBClusterSnapshot (ação)

O nome da CLI da AWS para essa API é: `delete-db-cluster-snapshot`.

Exclui um snapshot do cluster de banco de dados. Se o snapshot estiver sendo copiado, a operação de cópia será encerrada.

> **Note**
>
> O snapshot do cluster de banco de dados deve estar no estado `available` para ser excluído.

Solicitação

- DBClusterSnapshotIdentifier (na CLI: `--db-cluster-snapshot-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador do snapshot do cluster de banco de dados a ser excluído.

  Restrições: deve ser o nome de um snapshot do cluster de banco de dados existente no estado `available`.

Resposta

Contém os detalhes de um snapshot de cluster de banco de dados do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta na ação the section called "DescribeDBClusterSnapshots".

- AllocatedStorage: um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica o tamanho de armazenamento alocado em gibibytes (GiB).
- AvailabilityZones: uma string, do tipo: `string` (uma string codificada em UTF-8).

Fornece a lista de zonas de disponibilidade do EC2 nas quais as instâncias no snapshot de cluster de banco de dados podem ser restauradas.

- ClusterCreateTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a hora em que o cluster de banco de dados foi criado, no Tempo Universal Coordenado (UTC).

- DBClusterIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o identificador do cluster de banco de dados do qual esse snapshot de cluster de banco de dados foi criado.

- DBClusterSnapshotArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) do snapshot do cluster de banco de dados.

- DBClusterSnapshotIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o identificador de um snapshot de cluster de banco de dados. Deve corresponder ao identificador de um snapshot existente.

  Depois de restaurar um cluster de banco de dados usando um `DBClusterSnapshotIdentifier`, é necessário especificar o mesmo `DBClusterSnapshotIdentifier` para qualquer atualização futura para o cluster de banco de dados. Quando você especifica essa propriedade para uma atualização, o cluster de banco de dados não é restaurado do snapshot novamente, e os dados no banco de dados não são alterados.

  No entanto, se você não especificar o `DBClusterSnapshotIdentifier`, será criado um cluster de banco de dados vazio e o cluster de banco de dados original será excluído. Se você especificar uma propriedade diferente da propriedade de restauração de snapshot anterior, o cluster de banco de dados será restaurado do snapshot especificado pelo `DBClusterSnapshotIdentifier` e o cluster de banco de dados original será excluído.

- Engine: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o nome do mecanismo de banco de dados.

- EngineVersion: uma string, do tipo: `string` (uma string codificada em UTF-8).

Fornece a versão do mecanismo de banco de dados para esse snapshot do cluster de banco de dados.

- IAMDatabaseAuthenticationEnabled: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  True se o mapeamento de contas do Amazon Identity and Access Management (IAM) para contas de banco de dados estiver habilitado. Caso contrário, false.

- KmsKeyId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Se `StorageEncrypted` for verdadeiro, o identificador da chave do Amazon KMS para o cluster de banco de dados criptografado.

- LicenseModel: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece as informações do modelo de licença para esse snapshot do cluster de banco de dados.

- PercentProgress: um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica a porcentagem dos dados estimados transferidos.

- Port: um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica a porta na qual o cluster de banco de dados estava realizando a escuta no momento do snapshot.

- SnapshotCreateTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Fornece a hora em que o snapshot foi criado, no Tempo Universal Coordenado (UTC).

- SnapshotType: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o tipo do snapshot do cluster de banco de dados.

- SourceDBClusterSnapshotArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Se o snapshot do cluster de banco de dados foi copiado de um snapshot do cluster de banco de dados de origem, o nome de recurso da Amazon (ARN) do snapshot do cluster de banco de dados de origem, caso contrário, um valor nulo.

- Status: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o status desse snapshot de cluster de banco de dados.

- StorageEncrypted: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

Especifica se o snapshot do cluster de banco de dados está criptografado.

- StorageType: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo de armazenamento associado ao snapshot de cluster de banco de dados.

- VpcId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o ID da VPC associada ao snapshot de cluster de banco de dados.

Erros

- InvalidDBClusterSnapshotStateFault
- DBClusterSnapshotNotFoundFault

CopyDBClusterSnapshot (ação)

O nome da CLI da AWS para essa API é: `copy-db-cluster-snapshot`.

Copia um snapshot de um cluster de banco de dados.

Para copiar um snapshot de um snapshot manual do cluster de banco de dados compartilhado, `SourceDBClusterSnapshotIdentifier` deve ser o nome de recurso da Amazon (ARN) do snapshot do cluster de banco de dados compartilhado.

Solicitação

- CopyTags (na CLI: `--copy-tags`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  True para copiar todas as tags do snapshot do cluster de banco de dados de origem no snapshot do cluster de banco de dados de destino. Caso contrário, false. O padrão é falso.

- KmsKeyId (na CLI: `--kms-key-id`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da chave do Amazon KMS para um snapshot do cluster de banco de dados criptografado. O ID da chave KMS é o Nome de Recurso Amazon (ARN), o identificador da chave KMS ou o alias da chave KMS para a chave de criptografia do KMS.

Se você copiar um snapshot do cluster de banco de dados criptografado da conta da Amazon, poderá especificar um valor para `KmsKeyId` a fim de criptografar a cópia com uma nova chave de criptografia do KMS. Se você não especificar um valor para `KmsKeyId`, a cópia do snapshot do cluster de banco de dados será criptografada com a mesma chave do KMS que o snapshot do cluster de banco de dados de origem.

Se você copiar um snapshot de cluster de banco de dados criptografado compartilhado de outra conta da Amazon, deverá especificar um valor para `KmsKeyId`.

As chaves de criptografia do KMS são específicas das regiões da Amazon em que são criadas. Não é possível usar chaves de criptografia de uma região da Amazon em outra região da Amazon.

Você não pode criptografar um snapshot de cluster de banco de dados não criptografado ao copiá-lo. Se você tentar copiar um snapshot de cluster de banco de dados não criptografado e especificar um valor para o parâmetro KmsKeyId, um erro será retornado.

- PreSignedUrl (na CLI: `--pre-signed-url`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Sem suporte no momento.
- SourceDBClusterSnapshotIdentifier (na CLI: `--source-db-cluster-snapshot-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador do snapshot do cluster de banco de dados a ser copiado. Esse parâmetro não diferencia maiúsculas de minúsculas.

  Restrições:

  - É necessário especificar um snapshot de sistema válido no estado "disponível".
  - Especifique um identificador de DB snapshot.

  Exemplo: `my-cluster-snapshot1`
- Tags (na CLI: `--tags`): uma matriz de objetos Tag.

  As tags a serem atribuídas à nova cópia do snapshot do cluster de banco de dados.
- TargetDBClusterSnapshotIdentifier (na CLI: `--target-db-cluster-snapshot-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador do novo snapshot do cluster de banco de dados a ser criado a partir do snapshot do cluster de banco de dados de origem. Esse parâmetro não diferencia maiúsculas de minúsculas.

Restrições:

- Deve conter de 1 a 63 caracteres, incluindo letras, números ou hífens.
- O primeiro caractere deve ser uma letra.
- Não podem terminar com um hífen ou conter dois hífens consecutivos.

Exemplo: `my-cluster-snapshot2`

Resposta

Contém os detalhes de um snapshot de cluster de banco de dados do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta na ação the section called "DescribeDBClusterSnapshots".

- AllocatedStorage: um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica o tamanho de armazenamento alocado em gibibytes (GiB).
- AvailabilityZones: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece a lista de zonas de disponibilidade do EC2 nas quais as instâncias no snapshot de cluster de banco de dados podem ser restauradas.
- ClusterCreateTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a hora em que o cluster de banco de dados foi criado, no Tempo Universal Coordenado (UTC).
- DBClusterIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o identificador do cluster de banco de dados do qual esse snapshot de cluster de banco de dados foi criado.
- DBClusterSnapshotArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) do snapshot do cluster de banco de dados.
- DBClusterSnapshotIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o identificador de um snapshot de cluster de banco de dados. Deve corresponder ao identificador de um snapshot existente.

Depois de restaurar um cluster de banco de dados usando um `DBClusterSnapshotIdentifier`, é necessário especificar o mesmo `DBClusterSnapshotIdentifier` para qualquer atualização futura para o cluster de banco de dados. Quando você especifica essa propriedade para uma atualização, o cluster de banco de dados não é restaurado do snapshot novamente, e os dados no banco de dados não são alterados.

No entanto, se você não especificar o `DBClusterSnapshotIdentifier`, será criado um cluster de banco de dados vazio e o cluster de banco de dados original será excluído. Se você especificar uma propriedade diferente da propriedade de restauração de snapshot anterior, o cluster de banco de dados será restaurado do snapshot especificado pelo `DBClusterSnapshotIdentifier` e o cluster de banco de dados original será excluído.

- Engine: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o nome do mecanismo de banco de dados.

- EngineVersion: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece a versão do mecanismo de banco de dados para esse snapshot do cluster de banco de dados.

- IAMDatabaseAuthenticationEnabled: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  True se o mapeamento de contas do Amazon Identity and Access Management (IAM) para contas de banco de dados estiver habilitado. Caso contrário, false.

- KmsKeyId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Se `StorageEncrypted` for verdadeiro, o identificador da chave do Amazon KMS para o cluster de banco de dados criptografado.

- LicenseModel: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece as informações do modelo de licença para esse snapshot do cluster de banco de dados.

- PercentProgress: um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica a porcentagem dos dados estimados transferidos.

- Port: um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

Especifica a porta na qual o cluster de banco de dados estava realizando a escuta no momento do snapshot.

- SnapshotCreateTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Fornece a hora em que o snapshot foi criado, no Tempo Universal Coordenado (UTC).

- SnapshotType: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o tipo do snapshot do cluster de banco de dados.

- SourceDBClusterSnapshotArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Se o snapshot do cluster de banco de dados foi copiado de um snapshot do cluster de banco de dados de origem, o nome de recurso da Amazon (ARN) do snapshot do cluster de banco de dados de origem, caso contrário, um valor nulo.

- Status: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o status desse snapshot de cluster de banco de dados.

- StorageEncrypted: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se o snapshot do cluster de banco de dados está criptografado.

- StorageType: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo de armazenamento associado ao snapshot de cluster de banco de dados.

- VpcId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o ID da VPC associada ao snapshot de cluster de banco de dados.

Erros

- DBClusterSnapshotAlreadyExistsFault
- DBClusterSnapshotNotFoundFault
- InvalidDBClusterStateFault
- InvalidDBClusterSnapshotStateFault
- SnapshotQuotaExceededFault
- KMSKeyNotAccessibleFault

ModifyDBClusterSnapshotAttribute (ação)

O nome da CLI da AWS para essa API é: `modify-db-cluster-snapshot-attribute`.

Adiciona um atributo e os valores ou remove um atributo e os valores de um snapshot do cluster de banco de dados manual.

Para compartilhar um snapshot manual do cluster de banco de dados com outras contas da Amazon, especifique `restore` como `AttributeName` e use o parâmetro `ValuesToAdd` para adicionar uma lista de IDs das contas da Amazon que são autorizadas a restaurar o snapshot manual do cluster de banco de dados. Use o valor `all` para tornar público o snapshot manual do cluster de banco de dados, o que significa que ele pode ser copiado ou restaurado por todas as contas da Amazon. Não adicione o valor `all` a nenhum snapshot manual do cluster de banco de dados que contenha informações privadas que você não deseja disponibilizar para todas as contas da Amazon. Se um snapshot manual do cluster de banco de dados for criptografado, ele poderá ser compartilhado, mas apenas especificando uma lista de IDs de contas da Amazon autorizados para o parâmetro `ValuesToAdd`. Você não pode usar `all` como um valor para esse parâmetro nesse caso.

Para ver quais contas da Amazon têm acesso para copiar ou restaurar um snapshot manual do cluster de banco de dados ou se um snapshot manual do cluster de banco de dados é público ou privado, use a ação da API the section called “DescribeDBClusterSnapshotAttributes”.

Solicitação

- AttributeName (na CLI: `--attribute-name`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do atributo do snapshot do cluster de banco de dados a ser modificado.

  Para gerenciar a autorização de outras contas da Amazon para copiar ou restaurar um snapshot manual do cluster do banco de dados, defina esse valor como `restore`.
- DBClusterSnapshotIdentifier (na CLI: `--db-cluster-snapshot-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador do snapshot do cluster de banco de dados cujos atributos serão modificados.
- ValuesToAdd (na CLI: `--values-to-add`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de atributos de snapshot do cluster de banco de dados a serem adicionados ao atributo especificado por `AttributeName`.

Para autorizar outras contas da Amazon a copiar ou restaurar snapshot manual do cluster de banco de dados, defina essa lista para incluir um ou mais IDs de conta da Amazon, ou `all` para tornar o snapshot manual do cluster de banco de dados restaurável por qualquer conta da Amazon. Não adicione o valor `all` a nenhum snapshot manual do cluster de banco de dados que contenha informações privadas que você não deseja disponibilizar para todas as contas da Amazon.

- ValuesToRemove (na CLI: `--values-to-remove`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de atributos de snapshot do cluster de banco de dados a serem removidos do atributo especificado por `AttributeName`.

  Para remover a autorização de outras contas da Amazon para copiar ou restaurar um snapshot manual do cluster de banco de dados, defina essa lista para incluir um ou mais identificadores de conta da Amazon, ou `all` para remover a autorização de uma conta da Amazon para copiar ou restaurar o snapshot do cluster de banco de dados. Se você especificar `all`, uma conta da Amazon cujo ID for explicitamente adicionado ao atributo `restore` ainda poderá copiar ou restaurar um snapshot manual do cluster de banco de dados.

Resposta

Contém os resultados de uma chamada bem-sucedida para a ação da API the section called "DescribeDBClusterSnapshotAttributes".

Os atributos de um snapshot manual do cluster de banco de dados são usados para autorizar outras contas da Amazon a copiar ou restaurar um snapshot manual do cluster de banco de dados. Para obter mais informações, consulte Ação da API the section called "ModifyDBClusterSnapshotAttribute".

- DBClusterSnapshotAttributes – Uma matriz de objetos DBClusterSnapshotAttribute.

  A lista de atributos e valores do snapshot manual do cluster de banco de dados.

- DBClusterSnapshotIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador do snapshot manual do cluster de banco de dados ao qual os atributos se aplicam.

Erros

- DBClusterSnapshotNotFoundFault
- InvalidDBClusterSnapshotStateFault
- SharedSnapshotQuotaExceededFault

RestoreDBClusterFromSnapshot (ação)

O nome da CLI da AWS para essa API é: `restore-db-cluster-from-snapshot`.

Cria um novo cluster de banco de dados a partir de um DB snapshot ou de um snapshot de cluster de banco de dados.

Se um DB snapshot for especificado, o cluster de banco de dados de destino será criado a partir do DB snapshot de origem com uma configuração e um grupo de segurança padrão.

Se um snapshot do cluster de banco de dados for especificado, o cluster de banco de dados de destino será criado a partir do ponto de restauração do cluster de banco de dados de origem com a mesma configuração do cluster de banco de dados original, exceto pelo fato de que o novo cluster de banco de dados será criado com o grupo de segurança padrão.

Solicitação

- AvailabilityZones (na CLI: `--availability-zones`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece a lista de zonas de disponibilidade do EC2 nas quais as instâncias no snapshot de cluster de banco de dados restaurado podem ser criadas.
- CopyTagsToSnapshot (na CLI: `--copy-tags-to-snapshot`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se definido como *`true`*, as tags serão copiadas em qualquer snapshot do cluster de banco de dados restaurado que for criado.
- DatabaseName (na CLI: `--database-name`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Sem suporte.
- DBClusterIdentifier (na CLI: `--db-cluster-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

O nome do cluster de banco de dados a ser criado a partir do DB snapshot ou do snapshot do cluster de banco de dados. Este parâmetro não diferencia maiúsculas de minúsculas.

Restrições:

- Deve conter de 1 a 63 caracteres, incluindo letras, números ou hífens
- O primeiro caractere deve ser uma letra
- Não podem terminar com um hífen ou conter dois hífens consecutivos

Exemplo: `my-snapshot-id`

- DBClusterParameterGroupName (na CLI: `--db-cluster-parameter-group-name`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do grupo de parâmetros de cluster de banco de dados para associar a este novo cluster de banco de dados.

  Restrições:

  - Se for fornecido, deverá corresponder ao nome de um DBClusterParameterGroup existente.
- DBSubnetGroupName (na CLI: `--db-subnet-group-name`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do grupo de sub-redes de banco de dados a ser usado no novo cluster de banco de dados.

  Restrições: se fornecido, deve corresponder ao nome de um DBSubnetGroup existente.

  Exemplo: `mySubnetgroup`
- DeletionProtection (na CLI: `--deletion-protection`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Um valor que indica se o cluster de banco de dados tem a proteção contra exclusão habilitada. O banco de dados não pode ser excluído quando a proteção contra exclusão está habilitada. Por padrão, a proteção contra exclusão fica desabilitada.
- EnableCloudwatchLogsExports (na CLI: `--enable-cloudwatch-logs-exports`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  A lista de logs que o cluster de banco de dados restaurado deve exportar para o Amazon CloudWatch Logs.

- EnableIAMDatabaseAuthentication (na CLI: `--enable-iam-database-authentication`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Verdadeiro para habilitar o mapeamento de contas do Amazon Identity and Access Management (IAM) para contas de banco de dados. Caso contrário, falso.

  Padrão: `false`

- Engine (na CLI: `--engine`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O mecanismo de banco de dados a utilizar para o novo cluster de banco de dados.

  Padrão: o mesmo que a origem

  Restrição: deve ser compatível com o mecanismo da origem

- EngineVersion (na CLI: `--engine-version`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  A versão do mecanismo de banco de dados a ser usada para o novo cluster de banco de dados.

- KmsKeyId (na CLI: `--kms-key-id`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador da chave do Amazon KMS a ser usada ao restaurar um cluster de banco de dados criptografado a partir de um snapshot de banco de dados ou um snapshot do cluster de banco de dados.

  O identificador de chave KMS é o Amazon Resource Name (ARN) da chave de criptografia KMS. Se você estiver restaurando um cluster de banco de dados com a mesma conta da Amazon que tem a chave de criptografia do KMS usada para criptografar o novo cluster de banco de dados, use o alias da chave do KMS em vez do ARN para a chave de criptografia do KMS.

  Se você não especificar um valor para o parâmetro `KmsKeyId`, ocorrerá o seguinte:

  - Se o DB snapshot ou o snapshot do cluster de banco de dados em `SnapshotIdentifier` for criptografado, o cluster de banco de dados restaurado será criptografado usando a chave do KMS que foi utilizada para criptografar o DB snapshot ou o snapshot do cluster de banco de dados.
  - Se o DB snapshot ou do cluster de banco de dados em `SnapshotIdentifier` não estiver criptografado, o cluster de banco de dados restaurado não será criptografado.

- Port (na CLI: `--port`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número da porta na qual o novo cluster de banco de dados aceita conexões.

  Restrições: o valor deve ser `1150-65535`

  Padrão: a mesma porta que a do cluster de banco de dados original.

- ServerlessV2ScalingConfiguration (na CLI: `--serverless-v2-scaling-configuration`): um objeto ServerlessV2ScalingConfiguration.

  Contém a configuração de escalabilidade de um cluster de banco de dados do Neptune Serverless.

  Para obter mais informações, consulte Using Amazon Neptune Serverless no Guia do usuário do Amazon Neptune.

- SnapshotIdentifier (na CLI: `--snapshot-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador do qual restaurar o DB snapshot ou o snapshot do cluster de banco de dados.

  Você pode usar o nome ou o nome de recurso da Amazon (ARN) para especificar um snapshot do cluster de banco de dados. No entanto, é possível usar apenas o ARN para especificar um DB snapshot.

  Restrições:

  - Deve corresponder ao identificador de um snapshot existente.

- StorageType (na CLI: `--storage-type`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o tipo de armazenamento a ser associado ao cluster de banco de dados.

  Valores válidos: `standard`, `iopt1`

  Padrão: `standard`

- Tags (na CLI: `--tags`): uma matriz de objetos Tag.

  As tags a serem atribuídas ao cluster de banco de dados restaurado.

- VpcSecurityGroupIds (na CLI: `--vpc-security-group-ids`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  A lista de grupos de segurança da VPC a que o novo cluster de banco de dados pertencerá.

Resposta

Contém os detalhes de um cluster de banco de dados do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta em the section called "DescribeDBClusters".

- AllocatedStorage: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  `AllocatedStorage` sempre retorna 1, pois o tamanho de armazenamento de cluster de banco de dados do Neptune não é fixo, mas se ajusta automaticamente conforme necessário.
- AssociatedRoles – Uma matriz de objetos DBClusterRole.

  Fornece uma lista dos perfis do Amazon Identity and Access Management (IAM) que estão associadas ao cluster de banco de dados. Os perfis do IAM associados a uma permissão de concessão do cluster de banco de dados para o cluster de banco de dados para acessar outros serviços da Amazon em seu nome.
- AutomaticRestartTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Hora em que o cluster de banco de dados será reiniciado automaticamente.
- AvailabilityZones: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece a lista de zonas de disponibilidade do EC2 nas quais as instâncias no cluster de banco de dados podem ser criadas.
- BacktrackConsumedChangeRecords: um LongOptional, do tipo: `long` (um valor inteiro assinado de 64 bits).

  Não compatível com o Neptune.
- BacktrackWindow: um LongOptional, do tipo: `long` (um valor inteiro assinado de 64 bits).

  Não compatível com o Neptune.

- BackupRetentionPeriod: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica o número de dias durante os quais os DB snapshots automáticos são retidos.

- Capacity: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Não compatível com o Neptune.

- CloneGroupId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Identifica o grupo de clones ao qual o cluster do banco de dados está associado.

- ClusterCreateTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a hora em que o cluster de banco de dados foi criado, no Tempo Universal Coordenado (UTC).

- CopyTagsToSnapshot: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se definido como *`true`*, as tags serão copiadas em qualquer snapshot do cluster de banco de dados criado.

- CrossAccountClone: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se definido como `true`, o cluster de banco de dados poderá ser clonado em várias contas.

- DatabaseName: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém o nome do banco de dados inicial desse cluster de banco de dados que foi fornecido no momento da criação, caso tenha sido especificado quando o cluster de banco de dados foi criado. Esse mesmo nome é retornado durante toda a duração do cluster de banco de dados.

- DBClusterArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) do cluster de banco de dados.

- DBClusterIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um identificador de cluster de banco de dados fornecido pelo usuário. Esse identificador é a chave exclusiva que identifica um cluster de banco de dados.

- DBClusterMembers – Uma matriz de objetos DBClusterMember.

Fornece a lista de instâncias que compõem o cluster de banco de dados.

- DBClusterParameterGroup: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o nome do grupo de parâmetros de cluster de banco de dados para o cluster de banco de dados.

- DbClusterResourceId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador exclusivo e imutável da região da Amazon para o cluster de banco de dados. Esse identificador é encontrado nas entradas de log do Amazon CloudTrail sempre que a chave do Amazon KMS para o cluster de banco de dados é acessada.

- DBSubnetGroup: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica informações sobre o grupo de sub-redes associado ao cluster de banco de dados, incluindo o nome, a descrição e as sub-redes no grupo de sub-redes.

- DeletionProtection: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica se o cluster de banco de dados tem a proteção contra exclusão ativada ou não. O banco de dados não pode ser excluído quando a proteção contra exclusão está habilitada.

- EarliestBacktrackTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Não compatível com o Neptune.

- EarliestRestorableTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a data mais antiga para a qual um banco de dados pode ser restaurado com restauração point-in-time.

- EnabledCloudwatchLogsExports: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de tipos de logs que esse cluster de banco de dados está configurado para exportar para o CloudWatch Logs. Os tipos de log válidos são: `audit` (para publicar logs de auditoria no CloudWatch) e slowquery (para publicar logs de consulta lenta no CloudWatch). Consulte Publicar logs do Neptune no Amazon CloudWatch Logs.

- Endpoint: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o endpoint de conexão para a instância principal do cluster de banco de dados.

- Engine: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o nome do mecanismo de banco de dados a ser usado para esse cluster de banco de dados.
- EngineVersion: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Indica a versão do mecanismo do banco de dados.
- GlobalClusterIdentifier: um GlobalClusterIdentifier, do tipo: `string` (uma string codificada em UTF-8), no mínimo 1 e no máximo 255 ?st?s, correspondendo a esta expressão regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  Contém um identificador de cluster de banco de dados global fornecido pelo usuário. Esse identificador é a chave exclusiva que identifica um banco de dados global.
- HostedZoneId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o ID que o Amazon Route 53 atribui ao criar uma zona hospedada.
- IAMDatabaseAuthenticationEnabled: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  True se o mapeamento de contas do Amazon Identity and Access Management (IAM) para contas de banco de dados estiver habilitado. Caso contrário, false.
- IOOptimizedNextAllowedModificationTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Na próxima vez que for possível modificar o cluster de banco de dados para usar o tipo de armazenamento `iopt1`.
- KmsKeyId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Se `StorageEncrypted` for true, o identificador da chave Amazon KMS para o cluster de banco de dados criptografado.
- LatestRestorableTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a data mais recente até a qual um banco de dados pode ser restaurado com restauração pontual.
- MultiAZ: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se o cluster de banco de dados tem instâncias em várias zonas de disponibilidade.

- PendingModifiedValues: um objeto ClusterPendingModifiedValues.

  Esse tipo de dados é usado como um elemento de resposta na operação `ModifyDBCluster` e contém alterações que serão aplicadas durante a próxima janela de manutenção.

- PercentProgress: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o andamento da operação como uma porcentagem.

- Port: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica a porta onde o mecanismo de banco de dados está realizando a recepção.

- PreferredBackupWindow: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o intervalo de tempo diário durante o qual os backups automatizados serão criados se eles estiverem habilitados, conforme determinado por `BackupRetentionPeriod`.

- PreferredMaintenanceWindow: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o período semanal durante o qual pode ocorrer a manutenção do sistema, em Tempo Universal Coordenado (UTC).

- ReaderEndpoint: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O endpoint do leitor do cluster de banco de dados. O endpoint do leitor para uma carga do cluster de banco de dados equilibra as conexões entre as réplicas de leitura que estão disponíveis em um cluster de banco de dados. À medida que os clientes solicitam novas conexões ao endpoint do leitor, o Neptune distribui as solicitações de conexão entre as réplicas de leitura no cluster de banco de dados. Essa funcionalidade pode ajudar a equilibrar sua carga de trabalho de leitura entre várias réplicas de leitura em seu cluster de banco de dados.

  Se ocorrer um failover e a réplica de leitura à qual você estiver conectado for promovida à instância principal, sua conexão será interrompida. Para continuar a enviar sua carga de trabalho de leitura a outras réplicas de leitura no cluster, reconecte-se ao endpoint do leitor.

- ReadReplicaIdentifiers: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um ou mais identificadores das réplicas de leitura associadas a esse cluster de banco de dados.

- ReplicationSourceIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Não compatível com o Neptune.

- ReplicationType: uma string, do tipo: `string` (uma string codificada em UTF-8).

Não compatível com o Neptune.

- ServerlessV2ScalingConfiguration: um objeto ServerlessV2ScalingConfigurationInfo.

  Mostra a configuração de escalabilidade de um cluster de banco de dados do Neptune Serverless.

  Para obter mais informações, consulte Using Amazon Neptune Serverless no Guia do usuário do Amazon Neptune.

- Status: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o estado atual desse cluster de banco de dados.

- StorageEncrypted: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se o cluster de banco de dados é criptografado.

- StorageType: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo de armazenamento usado pelo cluster de banco de dados.

  Valores válidos:

  - **`standard`**: (o padrão). Fornece armazenamento econômico do banco de dados para aplicações com uso moderado a pequeno de E/S.
  - **`iopt1`**: habilita o armazenamento otimizado para E/S desenvolvido para atender às necessidades de workloads de grafos de uso intenso de E/S que exigem preços previsíveis com baixa latência de E/S e throughput de E/S consistente.

    O armazenamento otimizado para E/S do Neptune só está disponível a partir da versão 1.3.0.0 do mecanismo.

- VpcSecurityGroups – Uma matriz de objetos VpcSecurityGroupMembership.

  Fornece uma lista de grupos de segurança da VPC à qual o cluster de banco de dados pertence.

Erros

- DBClusterAlreadyExistsFault
- DBClusterQuotaExceededFault
- StorageQuotaExceededFault
- DBSubnetGroupNotFoundFault

- DBSnapshotNotFoundFault
- DBClusterSnapshotNotFoundFault
- InsufficientDBClusterCapacityFault
- InsufficientStorageClusterCapacityFault
- InvalidDBSnapshotStateFault
- InvalidDBClusterSnapshotStateFault
- StorageQuotaExceededFault
- InvalidVPCNetworkStateFault
- InvalidRestoreFault
- DBSubnetGroupNotFoundFault
- InvalidSubnet
- OptionGroupNotFoundFault
- KMSKeyNotAccessibleFault
- DBClusterParameterGroupNotFoundFault

RestoreDBClusterToPointInTime (ação)

O nome da CLI da AWS para essa API é: `restore-db-cluster-to-point-in-time`.

Restaura um cluster de banco de dados para um momento arbitrário. Os usuários podem restaurar para qualquer point-in-time antes de `LatestRestorableTime` por até `BackupRetentionPeriod` dias. O cluster de banco de dados de destino é criado a partir do cluster de banco de dados de origem com a mesma configuração do cluster de banco de dados original, exceto pelo fato de que o novo cluster de banco de dados será criado com o grupo de segurança de banco de dados padrão.

> **Note**
>
> Essa ação restaura apenas o cluster de banco de dados, não as instâncias desse cluster de banco de dados. É necessário invocar a ação the section called "CreateDBInstance" para criar instâncias de banco de dados para o cluster de banco de dados restaurado, especificando o identificador do cluster de banco de dados restaurado em `DBClusterIdentifier`. Você pode criar instâncias de banco de dados somente após a ação `RestoreDBClusterToPointInTime` tiver sido concluída e o cluster de banco de dados estiver disponível.

Solicitação

- DBClusterIdentifier (na CLI: `--db-cluster-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do novo cluster de banco de dados a ser criado.

  Restrições:

  - Deve conter de 1 a 63 caracteres, incluindo letras, números ou hífens
  - O primeiro caractere deve ser uma letra
  - Não podem terminar com um hífen ou conter dois hífens consecutivos
- DBClusterParameterGroupName (na CLI: `--db-cluster-parameter-group-name`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do grupo de parâmetros de cluster de banco de dados para associar a este novo cluster de banco de dados.

  Restrições:

  - Se for fornecido, deverá corresponder ao nome de um DBClusterParameterGroup existente.
- DBSubnetGroupName (na CLI: `--db-subnet-group-name`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do grupo de sub-redes de banco de dados a ser usado para o novo cluster de banco de dados.

  Restrições: se fornecido, deve corresponder ao nome de um DBSubnetGroup existente.

  Exemplo: `mySubnetgroup`
- DeletionProtection (na CLI: `--deletion-protection`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Um valor que indica se o cluster de banco de dados tem a proteção contra exclusão habilitada. O banco de dados não pode ser excluído quando a proteção contra exclusão está habilitada. Por padrão, a proteção contra exclusão fica desabilitada.
- EnableCloudwatchLogsExports (na CLI: `--enable-cloudwatch-logs-exports`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  A lista de logs que o cluster de banco de dados restaurado deve exportar para o CloudWatch Logs.

- EnableIAMDatabaseAuthentication (na CLI: `--enable-iam-database-authentication`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Verdadeiro para habilitar o mapeamento de contas do Amazon Identity and Access Management (IAM) para contas de banco de dados. Caso contrário, falso.

  Padrão: `false`

- KmsKeyId (na CLI: `--kms-key-id`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador da chave do Amazon KMS a ser usada ao restaurar um cluster de banco de dados criptografado a partir de um cluster de banco de dados criptografado.

  O identificador de chave KMS é o Amazon Resource Name (ARN) da chave de criptografia KMS. Se você estiver restaurando um cluster de banco de dados com a mesma conta da Amazon que tem a chave de criptografia do KMS usada para criptografar o novo cluster de banco de dados, use o alias da chave do KMS em vez do ARN para a chave de criptografia do KMS.

  Você pode restaurar para um novo cluster de banco de dados e criptografar o novo cluster com uma chave do KMS que é diferente da chave do KMS usada para criptografar o cluster de banco de dados de origem. O novo cluster de banco de dados é criptografado com a chave do KMS identificada pelo parâmetro `KmsKeyId`.

  Se você não especificar um valor para o parâmetro `KmsKeyId`, ocorrerá o seguinte:

  - Se o cluster de banco de dados for criptografado, o cluster de banco de dados restaurado será criptografado usando a chave do KMS que foi utilizada para criptografar o cluster de banco de dados de origem.
  - Se o cluster de banco de dados não estiver criptografado, o cluster de banco de dados restaurado não será criptografado.

  Se `DBClusterIdentifier` referir-se a um cluster de banco de dados não criptografado, a solicitação de restauração será rejeitada.

- Port (na CLI: `--port`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número da porta na qual o novo cluster de banco de dados aceita conexões.

  Restrições: o valor deve ser `1150-65535`

Padrão: a mesma porta que a do cluster de banco de dados original.

- RestoreToTime (na CLI: `--restore-to-time`): um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  A data e a hora para às quais restaurar o cluster de banco de dados.

  Valores válidos: o valor deve ser expresso no formato do Tempo Universal Coordenado (UTC)

  Restrições:

  - Devem ser anteriores ao último momento restaurável da instância de banco de dados
  - Devem ser especificadas se o parâmetro `UseLatestRestorableTime` não for especificado
  - Não podem ser especificadas se o parâmetro `UseLatestRestorableTime` for true
  - Não podem ser especificadas se o parâmetro `RestoreType` for `copy-on-write`.

  Exemplo: `2015-03-07T23:45:00Z`

- RestoreType (na CLI: `--restore-type`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo de restauração a ser realizada. Você pode especificar um dos seguintes valores:

  - `full-copy` - o novo cluster de banco de dados é restaurado como uma cópia completa do cluster de banco de dados de origem.
  - `copy-on-write` - o novo cluster de banco de dados é restaurado como um clone do cluster de banco de dados de origem.

  Se você não especificar um valor `RestoreType`, o novo cluster de banco de dados será restaurado como uma cópia completa do cluster de banco de dados de origem.

- ServerlessV2ScalingConfiguration (na CLI: `--serverless-v2-scaling-configuration`): um objeto ServerlessV2ScalingConfiguration.

  Contém a configuração de escalabilidade de um cluster de banco de dados do Neptune Serverless.

  Para obter mais informações, consulte Using Amazon Neptune Serverless no Guia do usuário do Amazon Neptune.

- SourceDBClusterIdentifier (na CLI: `--source-db-cluster-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

O identificador do cluster de banco de dados de origem do qual restaurar.

Restrições:

- Deve ser o identificador de um DBCluster existente.

- StorageType (na CLI: `--storage-type`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o tipo de armazenamento a ser associado ao cluster de banco de dados.

  Valores válidos: `standard`, `iopt1`

  Padrão: `standard`

- Tags (na CLI: `--tags`): uma matriz de objetos Tag.

  As tags a serem aplicadas ao cluster de banco de dados restaurado.

- UseLatestRestorableTime (na CLI: `--use-latest-restorable-time`): um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Um valor que é definido como `true` para restaurar o cluster de banco de dados para o horário mais recente de backup restaurável. Caso contrário, `false`.

  Padrão: `false`

  Restrições: não poderão ser especificadas se o parâmetro `RestoreToTime` for especificado.

- VpcSecurityGroupIds (na CLI: `--vpc-security-group-ids`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de grupos de segurança da VPC à qual o novo cluster de banco de dados pertence.

Resposta

Contém os detalhes de um cluster de banco de dados do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta em the section called "DescribeDBClusters".

- AllocatedStorage: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

`AllocatedStorage` sempre retorna 1, pois o tamanho de armazenamento de cluster de banco de dados do Neptune não é fixo, mas se ajusta automaticamente conforme necessário.

- AssociatedRoles – Uma matriz de objetos DBClusterRole.

  Fornece uma lista dos perfis do Amazon Identity and Access Management (IAM) que estão associadas ao cluster de banco de dados. Os perfis do IAM associados a uma permissão de concessão do cluster de banco de dados para o cluster de banco de dados para acessar outros serviços da Amazon em seu nome.

- AutomaticRestartTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Hora em que o cluster de banco de dados será reiniciado automaticamente.

- AvailabilityZones: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece a lista de zonas de disponibilidade do EC2 nas quais as instâncias no cluster de banco de dados podem ser criadas.

- BacktrackConsumedChangeRecords: um LongOptional, do tipo: `long` (um valor inteiro assinado de 64 bits).

  Não compatível com o Neptune.

- BacktrackWindow: um LongOptional, do tipo: `long` (um valor inteiro assinado de 64 bits).

  Não compatível com o Neptune.

- BackupRetentionPeriod: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica o número de dias durante os quais os DB snapshots automáticos são retidos.

- Capacity: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Não compatível com o Neptune.

- CloneGroupId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Identifica o grupo de clones ao qual o cluster do banco de dados está associado.

- ClusterCreateTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

Especifica a hora em que o cluster de banco de dados foi criado, no Tempo Universal Coordenado (UTC).

- CopyTagsToSnapshot: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se definido como *`true`*, as tags serão copiadas em qualquer snapshot do cluster de banco de dados criado.

- CrossAccountClone: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se definido como `true`, o cluster de banco de dados poderá ser clonado em várias contas.

- DatabaseName: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém o nome do banco de dados inicial desse cluster de banco de dados que foi fornecido no momento da criação, caso tenha sido especificado quando o cluster de banco de dados foi criado. Esse mesmo nome é retornado durante toda a duração do cluster de banco de dados.

- DBClusterArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) do cluster de banco de dados.

- DBClusterIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um identificador de cluster de banco de dados fornecido pelo usuário. Esse identificador é a chave exclusiva que identifica um cluster de banco de dados.

- DBClusterMembers – Uma matriz de objetos DBClusterMember.

  Fornece a lista de instâncias que compõem o cluster de banco de dados.

- DBClusterParameterGroup: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o nome do grupo de parâmetros de cluster de banco de dados para o cluster de banco de dados.

- DbClusterResourceId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador exclusivo e imutável da região da Amazon para o cluster de banco de dados. Esse identificador é encontrado nas entradas de log do Amazon CloudTrail sempre que a chave do Amazon KMS para o cluster de banco de dados é acessada.

- DBSubnetGroup: uma string, do tipo: `string` (uma string codificada em UTF-8).

Especifica informações sobre o grupo de sub-redes associado ao cluster de banco de dados, incluindo o nome, a descrição e as sub-redes no grupo de sub-redes.

- DeletionProtection: um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica se o cluster de banco de dados tem a proteção contra exclusão ativada ou não. O banco de dados não pode ser excluído quando a proteção contra exclusão está habilitada.

- EarliestBacktrackTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Não compatível com o Neptune.

- EarliestRestorableTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a data mais antiga para a qual um banco de dados pode ser restaurado com restauração point-in-time.

- EnabledCloudwatchLogsExports: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de tipos de logs que esse cluster de banco de dados está configurado para exportar para o CloudWatch Logs. Os tipos de log válidos são: `audit` (para publicar logs de auditoria no CloudWatch) e slowquery (para publicar logs de consulta lenta no CloudWatch). Consulte Publicar logs do Neptune no Amazon CloudWatch Logs.

- Endpoint: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o endpoint de conexão para a instância principal do cluster de banco de dados.

- Engine: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o nome do mecanismo de banco de dados a ser usado para esse cluster de banco de dados.

- EngineVersion: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Indica a versão do mecanismo do banco de dados.

- GlobalClusterIdentifier: um GlobalClusterIdentifier, do tipo: `string` (uma string codificada em UTF-8), no mínimo 1 e no máximo 255 ?st?s, correspondendo a esta expressão regular: `[A-Za-z][0-9A-Za-z-:._]*`.

Contém um identificador de cluster de banco de dados global fornecido pelo usuário. Esse identificador é a chave exclusiva que identifica um banco de dados global.

- HostedZoneId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o ID que o Amazon Route 53 atribui ao criar uma zona hospedada.

- IAMDatabaseAuthenticationEnabled: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  True se o mapeamento de contas do Amazon Identity and Access Management (IAM) para contas de banco de dados estiver habilitado. Caso contrário, false.

- IOOptimizedNextAllowedModificationTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Na próxima vez que for possível modificar o cluster de banco de dados para usar o tipo de armazenamento `iopt1`.

- KmsKeyId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Se `StorageEncrypted` for true, o identificador da chave Amazon KMS para o cluster de banco de dados criptografado.

- LatestRestorableTime: um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a data mais recente até a qual um banco de dados pode ser restaurado com restauração pontual.

- MultiAZ: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se o cluster de banco de dados tem instâncias em várias zonas de disponibilidade.

- PendingModifiedValues: um objeto ClusterPendingModifiedValues.

  Esse tipo de dados é usado como um elemento de resposta na operação `ModifyDBCluster` e contém alterações que serão aplicadas durante a próxima janela de manutenção.

- PercentProgress: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o andamento da operação como uma porcentagem.

- Port: um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica a porta onde o mecanismo de banco de dados está realizando a recepção.

- PreferredBackupWindow: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o intervalo de tempo diário durante o qual os backups automatizados serão criados se eles estiverem habilitados, conforme determinado por `BackupRetentionPeriod`.

- PreferredMaintenanceWindow: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o período semanal durante o qual pode ocorrer a manutenção do sistema, em Tempo Universal Coordenado (UTC).

- ReaderEndpoint: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O endpoint do leitor do cluster de banco de dados. O endpoint do leitor para uma carga do cluster de banco de dados equilibra as conexões entre as réplicas de leitura que estão disponíveis em um cluster de banco de dados. À medida que os clientes solicitam novas conexões ao endpoint do leitor, o Neptune distribui as solicitações de conexão entre as réplicas de leitura no cluster de banco de dados. Essa funcionalidade pode ajudar a equilibrar sua carga de trabalho de leitura entre várias réplicas de leitura em seu cluster de banco de dados.

  Se ocorrer um failover e a réplica de leitura à qual você estiver conectado for promovida à instância principal, sua conexão será interrompida. Para continuar a enviar sua carga de trabalho de leitura a outras réplicas de leitura no cluster, reconecte-se ao endpoint do leitor.

- ReadReplicaIdentifiers: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um ou mais identificadores das réplicas de leitura associadas a esse cluster de banco de dados.

- ReplicationSourceIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Não compatível com o Neptune.

- ReplicationType: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Não compatível com o Neptune.

- ServerlessV2ScalingConfiguration: um objeto ServerlessV2ScalingConfigurationInfo.

  Mostra a configuração de escalabilidade de um cluster de banco de dados do Neptune Serverless.

  Para obter mais informações, consulte Using Amazon Neptune Serverless no Guia do usuário do Amazon Neptune.

- Status: uma string, do tipo: `string` (uma string codificada em UTF-8).

Especifica o estado atual desse cluster de banco de dados.

- StorageEncrypted: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se o cluster de banco de dados é criptografado.

- StorageType: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo de armazenamento usado pelo cluster de banco de dados.

  Valores válidos:

  - **`standard`**: (o padrão). Fornece armazenamento econômico do banco de dados para aplicações com uso moderado a pequeno de E/S.
  - **`iopt1`**: habilita o armazenamento otimizado para E/S desenvolvido para atender às necessidades de workloads de grafos de uso intenso de E/S que exigem preços previsíveis com baixa latência de E/S e throughput de E/S consistente.

    O armazenamento otimizado para E/S do Neptune só está disponível a partir da versão 1.3.0.0 do mecanismo.

- VpcSecurityGroups – Uma matriz de objetos VpcSecurityGroupMembership.

  Fornece uma lista de grupos de segurança da VPC à qual o cluster de banco de dados pertence.

Erros

- DBClusterAlreadyExistsFault
- DBClusterNotFoundFault
- DBClusterQuotaExceededFault
- DBClusterSnapshotNotFoundFault
- DBSubnetGroupNotFoundFault
- InsufficientDBClusterCapacityFault
- InsufficientStorageClusterCapacityFault
- InvalidDBClusterSnapshotStateFault
- InvalidDBClusterStateFault
- InvalidDBSnapshotStateFault
- InvalidRestoreFault

- InvalidSubnet
- InvalidVPCNetworkStateFault
- KMSKeyNotAccessibleFault
- OptionGroupNotFoundFault
- StorageQuotaExceededFault
- DBClusterParameterGroupNotFoundFault

DescribeDBClusterSnapshots (ação)

O nome da CLI da AWS para essa API é: `describe-db-cluster-snapshots`.

Retorna informações sobre snapshots do cluster banco de dados. Essa ação de API dá suporte à paginação.

Solicitação

- DBClusterIdentifier (na CLI: `--db-cluster-identifier`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID do cluster de banco de dados para o qual recuperar a lista de snapshots de cluster de banco de dados. Esse parâmetro não pode ser usado em conjunto com o parâmetro `DBClusterSnapshotIdentifier`. Esse parâmetro não diferencia maiúsculas de minúsculas.

  Restrições:

  - Se fornecido, deverá corresponder ao identificador de um DBCluster existente.
- DBClusterSnapshotIdentifier (na CLI: `--db-cluster-snapshot-identifier`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um identificador de snapshot de cluster de banco de dados específico a ser descrito. Esse parâmetro não pode ser usado em conjunto com o parâmetro `DBClusterIdentifier`. Esse valor é armazenado como uma string em minúsculas.

  Restrições:

  - Se fornecido, deverá corresponder ao identificador de um DBClusterSnapshot existente.
  - Se esse identificador for uma captura de tela automatizada, o parâmetro `SnapshotType` também deverá ser especificado.
- Filters (na CLI: `--filters`): uma matriz de objetos Filtro.

Não há suporte para esse parâmetro atualmente.

- IncludePublic (na CLI: `--include-public`): um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Verdadeiro para incluir snapshots manuais do cluster de banco de dados que são públicos e podem ser copiados ou restaurados por qualquer conta da Amazon. Caso contrário, falso. O padrão é `false`. O padrão é falso.

  Você pode compartilhar um snapshot manual do cluster de banco de dados como público usando a ação de API the section called “ModifyDBClusterSnapshotAttribute”.

- IncludeShared (na CLI: `--include-shared`): um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Verdadeiro para incluir snapshots manuais do cluster de banco de dados compartilhados de outras contas da Amazon que essa conta da Amazon tem permissão para copiar ou restaurar. Caso contrário, falso. O padrão é `false`.

  É possível fornecer a uma conta da Amazon permissão para restaurar um snapshot manual do cluster de banco de dados de outra conta da Amazon com a ação da API the section called “ModifyDBClusterSnapshotAttribute”.

- Marker (na CLI: `--marker`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um token de paginação opcional fornecido por uma solicitação `DescribeDBClusterSnapshots` anterior. Se esse parâmetro for especificado, a resposta incluirá apenas os registros além do marcador, até o valor especificado por `MaxRecords`.

- MaxRecords (na CLI: `--max-records`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número máximo de registros a serem incluídos na resposta. Se existirem mais registros do que o valor `MaxRecords` especificado, um token de paginação chamado de marcador será incluído na resposta para que os resultados restantes possam ser recuperados.

  Padrão: 100

  Restrições: Mínimo 20, máximo 100.

- SnapshotType (na CLI: `--snapshot-type`): uma string, do tipo: `string` (uma string codificada em UTF-8).

O tipo de snapshots de cluster de banco de dados a ser retornado. Você pode especificar um dos seguintes valores:

- `automated`: exibe todos os snapshots do cluster de banco de dados que foram criados pelo Amazon Neptune automaticamente para minha conta da Amazon.
- `manual`: exibe todos os snapshots do cluster de banco de dados que foram criados por minha conta da Amazon.
- `shared`: gera todos os snapshots manuais do cluster de banco de dados que foram compartilhados para a minha conta da Amazon.
- `public` – retorna todos os snapshots do cluster de banco de dados que foram marcados como públicos.

Se você não especificar um valor `SnapshotType`, os snapshots de cluster de banco de dados automatizados e manuais serão retornados. Você pode incluir snapshots do cluster de banco de dados compartilhados com esses resultados configurando o parâmetro `IncludeShared` como `true`. Você pode incluir snapshots do cluster de banco de dados públicos com esses resultados configurando o parâmetro `IncludePublic` como `true`.

Os parâmetros `IncludeShared` e `IncludePublic` não se aplicam aos valores `SnapshotType` de `manual` ou `automated`. O parâmetro `IncludePublic` não se aplica quando `SnapshotType` está definido como `shared`. O parâmetro `IncludeShared` não se aplica quando `SnapshotType` está definido como `public`.

Resposta

- DBClusterSnapshots – Uma matriz de objetos DBClusterSnapshot.

  Fornece uma lista de snapshots do cluster de banco de dados para o usuário.

- Marker: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um token de paginação opcional fornecido por uma solicitação the section called "DescribeDBClusterSnapshots" anterior. Se esse parâmetro for especificado, a resposta incluirá apenas os registros além do marcador, até o valor especificado por `MaxRecords`.

Erros

- DBClusterSnapshotNotFoundFault

DescribeDBClusterSnapshotAttributes (ação)

O nome da CLI da AWS para essa API é: `describe-db-cluster-snapshot-attributes`.

Retorna uma lista de nomes e valores de atributos de snapshot do cluster de banco de dados de um snapshot manual do cluster de banco de dados.

Ao compartilhar snapshots com outras contas da Amazon, `DescribeDBClusterSnapshotAttributes` exibe o atributo `restore` e uma lista de IDs das contas da Amazon que são autorizadas a copiar ou restaurar o snapshot manual do cluster de banco de dados. Se `all` estiver incluído na lista de valores do atributo `restore`, o snapshot manual do cluster de banco de dados será público e poderá ser copiado ou restaurado por todas as contas da Amazon.

Para adicionar ou remover o acesso de uma conta da Amazon para copiar ou restaurar um snapshot manual do cluster de banco de dados ou tornar público ou privado um snapshot manual do cluster de banco de dados, use a ação da API the section called “ModifyDBClusterSnapshotAttribute”.

Solicitação

- DBClusterSnapshotIdentifier (na CLI: `--db-cluster-snapshot-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador do snapshot do cluster de banco de dados cujos atributos serão descritos.

Resposta

Contém os resultados de uma chamada bem-sucedida para a ação da API the section called “DescribeDBClusterSnapshotAttributes”.

Os atributos de um snapshot manual do cluster de banco de dados são usados para autorizar outras contas da Amazon a copiar ou restaurar um snapshot manual do cluster de banco de dados. Para obter mais informações, consulte Ação da API the section called “ModifyDBClusterSnapshotAttribute”.

- DBClusterSnapshotAttributes – Uma matriz de objetos DBClusterSnapshotAttribute.

  A lista de atributos e valores do snapshot manual do cluster de banco de dados.
- DBClusterSnapshotIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

O identificador do snapshot manual do cluster de banco de dados ao qual os atributos se aplicam.

Erros

- DBClusterSnapshotNotFoundFault

Estruturas:

DBClusterSnapshot (estrutura)

Contém os detalhes de um snapshot de cluster de banco de dados do Amazon Neptune.

Esse tipo de dados é usado como um elemento de resposta na ação the section called "DescribeDBClusterSnapshots".

Campos

- AllocatedStorage: é um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica o tamanho de armazenamento alocado em gibibytes (GiB).
- AvailabilityZones: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece a lista de zonas de disponibilidade do EC2 nas quais as instâncias no snapshot de cluster de banco de dados podem ser restauradas.
- ClusterCreateTime: é um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a hora em que o cluster de banco de dados foi criado, no Tempo Universal Coordenado (UTC).
- DBClusterIdentifier: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o identificador do cluster de banco de dados do qual esse snapshot de cluster de banco de dados foi criado.
- DBClusterSnapshotArn: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) do snapshot do cluster de banco de dados.
- DBClusterSnapshotIdentifier: é uma string, do tipo: `string` (uma string codificada em UTF-8).

Especifica o identificador de um snapshot de cluster de banco de dados. Deve corresponder ao identificador de um snapshot existente.

Depois de restaurar um cluster de banco de dados usando um `DBClusterSnapshotIdentifier`, é necessário especificar o mesmo `DBClusterSnapshotIdentifier` para qualquer atualização futura para o cluster de banco de dados. Quando você especifica essa propriedade para uma atualização, o cluster de banco de dados não é restaurado do snapshot novamente, e os dados no banco de dados não são alterados.

No entanto, se você não especificar o `DBClusterSnapshotIdentifier`, será criado um cluster de banco de dados vazio e o cluster de banco de dados original será excluído. Se você especificar uma propriedade diferente da propriedade de restauração de snapshot anterior, o cluster de banco de dados será restaurado do snapshot especificado pelo `DBClusterSnapshotIdentifier` e o cluster de banco de dados original será excluído.

- Engine: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o nome do mecanismo de banco de dados.

- EngineVersion: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece a versão do mecanismo de banco de dados para esse snapshot do cluster de banco de dados.

- IAMDatabaseAuthenticationEnabled: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  True se o mapeamento de contas do Amazon Identity and Access Management (IAM) para contas de banco de dados estiver habilitado. Caso contrário, false.

- KmsKeyId: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Se `StorageEncrypted` for verdadeiro, o identificador da chave do Amazon KMS para o cluster de banco de dados criptografado.

- LicenseModel: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece as informações do modelo de licença para esse snapshot do cluster de banco de dados.

- PercentProgress: é um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica a porcentagem dos dados estimados transferidos.

- Port: é um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica a porta na qual o cluster de banco de dados estava realizando a escuta no momento do snapshot.

- SnapshotCreateTime: é um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Fornece a hora em que o snapshot foi criado, no Tempo Universal Coordenado (UTC).

- SnapshotType: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o tipo do snapshot do cluster de banco de dados.

- SourceDBClusterSnapshotArn: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Se o snapshot do cluster de banco de dados foi copiado de um snapshot do cluster de banco de dados de origem, o nome de recurso da Amazon (ARN) do snapshot do cluster de banco de dados de origem, caso contrário, um valor nulo.

- Status: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o status desse snapshot de cluster de banco de dados.

- StorageEncrypted: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Especifica se o snapshot do cluster de banco de dados está criptografado.

- StorageType: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo de armazenamento associado ao snapshot de cluster de banco de dados.

- VpcId: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o ID da VPC associada ao snapshot de cluster de banco de dados.

`DBClusterSnapshot` é usado como o elemento de resposta para:

- CreateDBClusterSnapshot
- CopyDBClusterSnapshot
- DeleteDBClusterSnapshot

DBClusterSnapshotAttribute (estrutura)

Contém o nome e os valores de um atributo de snapshot do cluster de banco de dados.

Os atributos de um snapshot manual do cluster de banco de dados são usados para autorizar outras contas da Amazon a restaurar um snapshot manual do cluster de banco de dados. Para obter mais informações, consulte Ação da API the section called “ModifyDBClusterSnapshotAttribute”.

Campos

- AttributeName: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do atributo do snapshot manual do cluster de banco de dados.

  O atributo chamado `restore` refere-se à lista de contas da Amazon que têm permissão para copiar ou restaurar o snapshot manual do cluster de banco de dados. Para obter mais informações, consulte Ação da API the section called “ModifyDBClusterSnapshotAttribute”.
- AttributeValues: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Os valores do atributo do snapshot manual do cluster de banco de dados.

  Se o campo `AttributeName` for definido como `restore`, esse elemento vai gerar uma lista de IDs das contas da Amazon que são autorizadas a copiar ou restaurar o snapshot manual do cluster de banco de dados. Se um valor de `all` estiver na lista, o snapshot manual do cluster de banco de dados será público e estará disponível para qualquer conta da Amazon copiar ou restaurar.

DBClusterSnapshotAttributesResult (estrutura)

Contém os resultados de uma chamada bem-sucedida para a ação da API the section called “DescribeDBClusterSnapshotAttributes”.

Os atributos de um snapshot manual do cluster de banco de dados são usados para autorizar outras contas da Amazon a copiar ou restaurar um snapshot manual do cluster de banco de dados. Para obter mais informações, consulte Ação da API the section called “ModifyDBClusterSnapshotAttribute”.

Campos

- DBClusterSnapshotAttributes: é uma matriz de objetos DBClusterSnapshotAttribute.

A lista de atributos e valores do snapshot manual do cluster de banco de dados.

- DBClusterSnapshotIdentifier: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador do snapshot manual do cluster de banco de dados ao qual os atributos se aplicam.

`DBClusterSnapshotAttributesResult` é usado como o elemento de resposta para:

- DescribeDBClusterSnapshotAttributes
- ModifyDBClusterSnapshotAttribute

API de eventos do Neptune

Ações:

- CreateEventSubscription (ação)
- DeleteEventSubscription (ação)
- ModifyEventSubscription (ação)
- DescribeEventSubscriptions (ação)
- AddSourceIdentifierToSubscription (ação)
- RemoveSourceIdentifierFromSubscription (ação)
- DescribeEvents (ação)
- DescribeEventCategories (ação)

Estruturas:

- Evento (estrutura)
- EventCategoriesMap (estrutura)
- EventSubscription (estrutura)

CreateEventSubscription (ação)

O nome da CLI da AWS para essa API é: `create-event-subscription`.

Cria uma assinatura de notificações de eventos. Essa ação exige um ARN de tópico (nome de recurso da Amazon) criado pelo console do Netuno, pelo console do SNS ou pela API do SNS. Para obter um ARN com o SNS, crie um tópico no Amazon SNS e assine-o. O ARN é exibido no console do SNS.

Você pode especificar o tipo de origem (SourceType) sobre o qual deseja ser notificado, fornecer uma lista de origens do Neptune (SourceIds) que aciona os eventos e fornecer uma lista de categorias de eventos (EventCategories) para eventos sobre os quais deseja ser notificado.
Por exemplo, você pode especificar SourceType = db-instance, SourceIds = mydbinstance1, mydbinstance2 e EventCategories = Availability, Backup.

Se especificar SourceType e SourceIds, como SourceType = db-instance e SourceIdentifier = myDBInstance1, você será notificado sobre todos os eventos db-instance para a origem especificada. Se você especificar um SourceType, mas não especificar um SourceIdentifier, você receberá um aviso dos eventos desse tipo de origem para todas as suas origens do Neptune.
Se você não especificar o SourceType nem o SourceIdentifier, receberá notificações de eventos gerados de todas as origens do Neptune que pertencem à sua conta de cliente.

Solicitação

- Enabled (na CLI: `--enabled`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Um valor booliano; definido como true para ativar a assinatura, definido como false para criar a assinatura, mas não ativá-la.
- EventCategories (na CLI: `--event-categories`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de categorias de evento para um SourceType que você deseja assinar. Você pode ver uma lista das categorias para um SourceType usando a ação DescribeEventCategories.
- SnsTopicArn (na CLI: `--sns-topic-arn`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) do tópico do SNS criado para notificação de eventos. O ARN é criado pelo Amazon SNS quando você cria um tópico e o assina.
- SourceIds (na CLI: `--source-ids`): uma string, do tipo: `string` (uma string codificada em UTF-8).

A lista de identificadores das origens de eventos para as quais os eventos são retornados. Se não for especificado, todas as origens serão incluídas na resposta. Um identificador deve começar com uma letra e conter apenas letras ASCII, dígitos e hifens e não terminar com um hífen nem conter dois hifens consecutivos.

Restrições:

- Se SourceIds forem fornecidos, o SourceType também deverá ser fornecido.
- Se o tipo de origem for uma instância de banco de dados, um `DBInstanceIdentifier` deverá ser fornecido.
- Se o tipo de origem for um grupo de segurança de banco de dados, um `DBSecurityGroupName` deverá ser fornecido.
- Se o tipo de origem for um grupo de parâmetros de banco de dados, um `DBParameterGroupName` deverá ser fornecido.
- Se o tipo de origem for um DB snapshot, um `DBSnapshotIdentifier` deverá ser fornecido.

- SourceType (na CLI: `--source-type`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo de origem que está gerando os eventos. Por exemplo, se você deseja ser notificado de eventos gerados por uma instância de banco de dados, defina esse parâmetro como db-instance. Se esse valor não for especificado, todos os eventos serão retornados.

  Valores válidos: `db-instance` | `db-cluster` | `db-parameter-group` | `db-security-group` | `db-snapshot` | `db-cluster-snapshot`

- SubscriptionName (na CLI: `--subscription-name`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome da inscrição.

  Restrições: o nome deve ter menos de 255 caracteres.

- Tags (na CLI: `--tags`): uma matriz de objetos Tag.

  As tags a serem aplicadas à nova assinatura de evento.

Resposta

Contém os resultados de uma invocação bem-sucedida da ação the section called "DescribeEventSubscriptions".

- CustomerAwsId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  A conta de cliente da Amazon associada à assinatura de notificações de eventos.
- CustSubscriptionId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID de assinatura de notificações de eventos.
- Enabled: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Um valor booliano que indica se a assinatura está ativada. True indica que a assinatura está ativada.
- EventCategoriesList: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de categorias de eventos da assinatura de notificações de eventos.
- EventSubscriptionArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) da assinatura de eventos.
- SnsTopicArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN do tópico da assinatura de notificações de eventos.
- SourceIdsList: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de IDs de origem da assinatura de notificações de eventos.
- SourceType: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo de origem da assinatura de notificações de eventos.
- Status: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O status da assinatura de notificações de eventos do .

  Restrições:

  Pode ser um dos seguintes: creating | modifying | deleting | active | no-permission | topic-not-exist

  O status "no-permission" indica que Neptune não tem mais permissão para fazer publicações no tópico do SNS. O status "topic-not-exist" indica que o tópico foi excluído após a assinatura ser criada.

- SubscriptionCreationTime: uma string, do tipo: `string` (uma string codificada em UTF-8).

  A hora em que a assinatura de notificações de eventos foi criada.

Erros

- EventSubscriptionQuotaExceededFault
- SubscriptionAlreadyExistFault
- SNSInvalidTopicFault
- SNSNoAuthorizationFault
- SNSTopicArnNotFoundFault
- SubscriptionCategoryNotFoundFault
- SourceNotFoundFault

DeleteEventSubscription (ação)

O nome da CLI da AWS para essa API é: `delete-event-subscription`.

Exclui uma assinatura de notificações de eventos.

Solicitação

- SubscriptionName (na CLI: `--subscription-name`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome da assinatura de notificações de eventos que você deseja excluir.

Resposta

Contém os resultados de uma invocação bem-sucedida da ação the section called "DescribeEventSubscriptions".

- CustomerAwsId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  A conta de cliente da Amazon associada à assinatura de notificações de eventos.
- CustSubscriptionId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID de assinatura de notificações de eventos.

- Enabled: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Um valor booliano que indica se a assinatura está ativada. True indica que a assinatura está ativada.

- EventCategoriesList: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de categorias de eventos da assinatura de notificações de eventos.

- EventSubscriptionArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) da assinatura de eventos.

- SnsTopicArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN do tópico da assinatura de notificações de eventos.

- SourceIdsList: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de IDs de origem da assinatura de notificações de eventos.

- SourceType: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo de origem da assinatura de notificações de eventos.

- Status: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O status da assinatura de notificações de eventos do .

  Restrições:

  Pode ser um dos seguintes: creating | modifying | deleting | active | no-permission | topic-not-exist

  O status "no-permission" indica que Neptune não tem mais permissão para fazer publicações no tópico do SNS. O status "topic-not-exist" indica que o tópico foi excluído após a assinatura ser criada.

- SubscriptionCreationTime: uma string, do tipo: `string` (uma string codificada em UTF-8).

  A hora em que a assinatura de notificações de eventos foi criada.

Erros

- SubscriptionNotFoundFault
- InvalidEventSubscriptionStateFault

ModifyEventSubscription (ação)

O nome da CLI da AWS para essa API é: `modify-event-subscription`.

Modifica uma assinatura de notificações de eventos existente. Não é possível modificar os identificadores de origem usando essa chamada; para alterar os identificadores de origem para uma assinatura, use as chamadas the section called “AddSourceIdentifierToSubscription” e the section called “RemoveSourceIdentifierFromSubscription”.

Você pode ver uma lista das categorias de eventos para um SourceType usando a ação DescribeEventCategories.

Solicitação

- Enabled (na CLI: `--enabled`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Um valor booliano; definido como true para ativar a assinatura.
- EventCategories (na CLI: `--event-categories`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de categorias de evento para um SourceType que você deseja assinar. Você pode ver uma lista das categorias para um SourceType usando a ação DescribeEventCategories.
- SnsTopicArn (na CLI: `--sns-topic-arn`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) do tópico do SNS criado para notificação de eventos. O ARN é criado pelo Amazon SNS quando você cria um tópico e o assina.
- SourceType (na CLI: `--source-type`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo de origem que está gerando os eventos. Por exemplo, se você deseja ser notificado de eventos gerados por uma instância de banco de dados, defina esse parâmetro como db-instance. Se esse valor não for especificado, todos os eventos serão retornados.

  Valores válidos: db-instance | db-parameter-group | db-security-group | db-snapshot
- SubscriptionName (na CLI: `--subscription-name`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

O nome da assinatura de notificações de eventos do .

Resposta

Contém os resultados de uma invocação bem-sucedida da ação the section called "DescribeEventSubscriptions".

- CustomerAwsId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  A conta de cliente da Amazon associada à assinatura de notificações de eventos.
- CustSubscriptionId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID de assinatura de notificações de eventos.
- Enabled: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Um valor booliano que indica se a assinatura está ativada. True indica que a assinatura está ativada.
- EventCategoriesList: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de categorias de eventos da assinatura de notificações de eventos.
- EventSubscriptionArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) da assinatura de eventos.
- SnsTopicArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN do tópico da assinatura de notificações de eventos.
- SourceIdsList: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de IDs de origem da assinatura de notificações de eventos.
- SourceType: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo de origem da assinatura de notificações de eventos.
- Status: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O status da assinatura de notificações de eventos do .

  Restrições:

Pode ser um dos seguintes: creating | modifying | deleting | active | no-permission | topic-not-exist

O status "no-permission" indica que Neptune não tem mais permissão para fazer publicações no tópico do SNS. O status "topic-not-exist" indica que o tópico foi excluído após a assinatura ser criada.

- SubscriptionCreationTime: uma string, do tipo: `string` (uma string codificada em UTF-8).

  A hora em que a assinatura de notificações de eventos foi criada.

Erros

- EventSubscriptionQuotaExceededFault
- SubscriptionNotFoundFault
- SNSInvalidTopicFault
- SNSNoAuthorizationFault
- SNSTopicArnNotFoundFault
- SubscriptionCategoryNotFoundFault

DescribeEventSubscriptions (ação)

O nome da CLI da AWS para essa API é: `describe-event-subscriptions`.

Lista todas as descrições de assinaturas de uma conta de cliente. A descrição de uma assinatura inclui SubscriptionName, SNSTopicARN, CustomerID, SourceType, SourceID, CreationTime e Status.

Se você especificar um SubscriptionName, listara a descrição dessa assinatura.

Solicitação

- Filters (na CLI: `--filters`): uma matriz de objetos Filtro.

  Não há suporte atualmente para esse parâmetro.
- Marker (na CLI: `--marker`): uma string, do tipo: `string` (uma string codificada em UTF-8).

Um token de paginação opcional fornecido por uma solicitação DescribeOrderableDBInstanceOptions anterior. Se esse parâmetro for especificado, a resposta incluirá apenas os registros além do marcador, até o valor especificado por `MaxRecords`.

- MaxRecords (na CLI: `--max-records`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número máximo de registros a serem incluídos na resposta. Se existirem mais registros do que o valor `MaxRecords` especificado, um token de paginação chamado de marcador será incluído na resposta para que os resultados restantes possam ser recuperados.

  Padrão: 100

  Restrições: mínimo 20, máximo 100.

- SubscriptionName (na CLI: `--subscription-name`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome da assinatura de notificações de eventos que você deseja assinar.

Resposta

- EventSubscriptionsList – Uma matriz de objetos EventSubscription.

  Uma lista de tipos de dados EventSubscriptions.

- Marker: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um token de paginação opcional fornecido por uma solicitação DescribeOrderableDBInstanceOptions anterior. Se esse parâmetro for especificado, a resposta incluirá apenas os registros além do marcador, até o valor especificado por `MaxRecords`.

Erros

- SubscriptionNotFoundFault

AddSourceIdentifierToSubscription (ação)

O nome da CLI da AWS para essa API é: `add-source-identifier-to-subscription`.

Adiciona um identificador de origem a uma assinatura de notificações de eventos existente.

Solicitação

- SourceIdentifier (na CLI: `--source-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador da origem do evento a ser adicionada.

  Restrições:

  - Se o tipo de origem for uma instância de banco de dados, um `DBInstanceIdentifier` deverá ser fornecido.
  - Se o tipo de origem for um grupo de segurança de banco de dados, um `DBSecurityGroupName` deverá ser fornecido.
  - Se o tipo de origem for um grupo de parâmetros de banco de dados, um `DBParameterGroupName` deverá ser fornecido.
  - Se o tipo de origem for um DB snapshot, um `DBSnapshotIdentifier` deverá ser fornecido.
- SubscriptionName (na CLI: `--subscription-name`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome da assinatura de notificações de eventos à qual você deseja adicionar um identificador de origem.

Resposta

Contém os resultados de uma invocação bem-sucedida da ação the section called “DescribeEventSubscriptions”.

- CustomerAwsId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  A conta de cliente da Amazon associada à assinatura de notificações de eventos.
- CustSubscriptionId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID de assinatura de notificações de eventos.
- Enabled: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Um valor booliano que indica se a assinatura está ativada. True indica que a assinatura está ativada.

- EventCategoriesList: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de categorias de eventos da assinatura de notificações de eventos.

- EventSubscriptionArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) da assinatura de eventos.

- SnsTopicArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN do tópico da assinatura de notificações de eventos.

- SourceIdsList: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de IDs de origem da assinatura de notificações de eventos.

- SourceType: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo de origem da assinatura de notificações de eventos.

- Status: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O status da assinatura de notificações de eventos do .

  Restrições:

  Pode ser um dos seguintes: creating | modifying | deleting | active | no-permission | topic-not-exist

  O status "no-permission" indica que Neptune não tem mais permissão para fazer publicações no tópico do SNS. O status "topic-not-exist" indica que o tópico foi excluído após a assinatura ser criada.

- SubscriptionCreationTime: uma string, do tipo: `string` (uma string codificada em UTF-8).

  A hora em que a assinatura de notificações de eventos foi criada.

Erros

- SubscriptionNotFoundFault
- SourceNotFoundFault

RemoveSourceIdentifierFromSubscription (ação)

O nome da CLI da AWS para essa API é: `remove-source-identifier-from-subscription`.

Remove um identificador de origem de uma assinatura de notificações de eventos existente.

Solicitação

- SourceIdentifier (na CLI: `--source-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador de origem a ser removido da assinatura, como o identificador de instância de banco de dados para uma instância de banco de dados ou o nome de um grupo de segurança.
- SubscriptionName (na CLI: `--subscription-name`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome da assinatura de notificações de eventos da qual você deseja remover um identificador de origem.

Resposta

Contém os resultados de uma invocação bem-sucedida da ação the section called "DescribeEventSubscriptions".

- CustomerAwsId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  A conta de cliente da Amazon associada à assinatura de notificações de eventos.
- CustSubscriptionId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID de assinatura de notificações de eventos.
- Enabled: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Um valor booliano que indica se a assinatura está ativada. True indica que a assinatura está ativada.
- EventCategoriesList: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de categorias de eventos da assinatura de notificações de eventos.
- EventSubscriptionArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

O nome de recurso da Amazon (ARN) da assinatura de eventos.

- SnsTopicArn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN do tópico da assinatura de notificações de eventos.

- SourceIdsList: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de IDs de origem da assinatura de notificações de eventos.

- SourceType: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo de origem da assinatura de notificações de eventos.

- Status: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O status da assinatura de notificações de eventos do .

  Restrições:

  Pode ser um dos seguintes: creating | modifying | deleting | active | no-permission | topic-not-exist

  O status "no-permission" indica que Neptune não tem mais permissão para fazer publicações no tópico do SNS. O status "topic-not-exist" indica que o tópico foi excluído após a assinatura ser criada.

- SubscriptionCreationTime: uma string, do tipo: `string` (uma string codificada em UTF-8).

  A hora em que a assinatura de notificações de eventos foi criada.

Erros

- SubscriptionNotFoundFault
- SourceNotFoundFault

DescribeEvents (ação)

O nome da CLI da AWS para essa API é: `describe-events`.

Retorna os eventos relacionados a instâncias de bancos de dados, grupos de segurança de banco de dados, DB snapshots e grupos de parâmetros de banco de dados dos últimos 14 dias. É possível obter eventos específicos de uma instância, um grupo de segurança, um snapshot ou um grupo de

parâmetros de banco de dados fornecendo o nome como um parâmetro. Por padrão, a última hora de eventos é retornada.

Solicitação

- Duration (na CLI: `--duration`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número de minutos dos quais recuperar eventos.

  Padrão: 60
- EndTime (na CLI: `--end-time`): um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  O fim do intervalo de tempo do qual recuperar eventos, especificado no formato ISO 8601. Para obter mais informações sobre ISO 8601, vá até a página da Wikipedia sobre ISO8601.

  Exemplo: 2009-07-08T18:00Z
- EventCategories (na CLI: `--event-categories`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de categorias que disparam notificações para uma assinatura de notificações de eventos.
- Filters (na CLI: `--filters`): uma matriz de objetos Filtro.

  Não há suporte atualmente para esse parâmetro.
- Marker (na CLI: `--marker`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um token de paginação opcional fornecido por uma solicitação DescribeEvents anterior. Se esse parâmetro for especificado, a resposta incluirá apenas os registros além do marcador, até o valor especificado por `MaxRecords`.
- MaxRecords (na CLI: `--max-records`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número máximo de registros a serem incluídos na resposta. Se existirem mais registros do que o valor `MaxRecords` especificado, um token de paginação chamado de marcador será incluído na resposta para que os resultados restantes possam ser recuperados.

  Padrão: 100

Restrições: mínimo 20, máximo 100.

- SourceIdentifier (na CLI: `--source-identifier`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador da origem do evento para o qual os eventos são retornados. Se não for especificado, todas as origens serão incluídas na resposta.

  Restrições:

  - Se SourceIdentifier for fornecido, o SourceType também deverá ser fornecido.
  - Se o tipo de origem for `DBInstance`, um `DBInstanceIdentifier` deverá ser fornecido.
  - Se o tipo de origem for `DBSecurityGroup`, um `DBSecurityGroupName` deverá ser fornecido.
  - Se o tipo de origem for `DBParameterGroup`, um `DBParameterGroupName` deverá ser fornecido.
  - Se o tipo de origem for `DBSnapshot`, um `DBSnapshotIdentifier` deverá ser fornecido.
  - Não podem terminar com um hífen ou conter dois hífens consecutivos.
- SourceType (na CLI: `--source-type`): um SourceType, do tipo: `string` (uma string codificada em UTF-8).

  A origem do evento da qual recuperar eventos. Se nenhum valor for especificado, todos os eventos serão retornados.
- StartTime (na CLI: `--start-time`): um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  O início do intervalo de tempo do qual recuperar eventos, especificado no formato ISO 8601. Para obter mais informações sobre ISO 8601, vá até a página da Wikipedia sobre ISO8601.

  Exemplo: 2009-07-08T18:00Z

Resposta

- Events – Uma matriz de objetos Evento.

  Uma lista de instâncias the section called “Evento”.
- Marker: uma string, do tipo: `string` (uma string codificada em UTF-8).

Um token de paginação opcional fornecido por uma solicitação Events. Se esse parâmetro for especificado, a resposta incluirá apenas os registros além do marcador, até o valor especificado por `MaxRecords`.

DescribeEventCategories (ação)

O nome da CLI da AWS para essa API é: `describe-event-categories`.

Exibe uma lista de categorias de todos os tipos de origem de eventos ou, se especificado, de um determinado tipo de origem.

Solicitação

- Filters (na CLI: `--filters`): uma matriz de objetos Filtro.

  Não há suporte atualmente para esse parâmetro.
- SourceType (na CLI: `--source-type`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo de origem que está gerando os eventos.

  Valores válidos: db-instance | db-parameter-group | db-security-group | db-snapshot

Resposta

- EventCategoriesMapList – Uma matriz de objetos EventCategoriesMap.

  Uma lista de tipos de dados EventCategoriesMap.

Estruturas:

Evento (estrutura)

Esse tipo de dados é usado como um elemento de resposta na ação the section called “DescribeEvents”.

Campos

- Date: é um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  Especifica a data e a hora do evento.
- EventCategories: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica a categoria do evento.
- Message: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o texto desse evento.
- SourceArn: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) do evento.
- SourceIdentifier: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Fornece o identificador para a origem do evento.
- SourceType: é um SourceType, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o tipo de origem desse evento.

EventCategoriesMap (estrutura)

Contém os resultados de uma invocação bem-sucedida da ação the section called "DescribeEventCategories".

Campos

- EventCategories: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  As categorias de eventos para o tipo de origem especificado
- SourceType: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo de origem ao qual pertencem as categorias retornadas

EventSubscription (estrutura)

Contém os resultados de uma invocação bem-sucedida da ação the section called "DescribeEventSubscriptions".

Campos

- CustomerAwsId: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  A conta de cliente da Amazon associada à assinatura de notificações de eventos.
- CustSubscriptionId: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID de assinatura de notificações de eventos.
- Enabled: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Um valor booliano que indica se a assinatura está ativada. True indica que a assinatura está ativada.
- EventCategoriesList: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de categorias de eventos da assinatura de notificações de eventos.
- EventSubscriptionArn: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) da assinatura de eventos.
- SnsTopicArn: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN do tópico da assinatura de notificações de eventos.
- SourceIdsList: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de IDs de origem da assinatura de notificações de eventos.
- SourceType: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo de origem da assinatura de notificações de eventos.
- Status: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O status da assinatura de notificações de eventos do .

  Restrições:

  Pode ser um dos seguintes: creating | modifying | deleting | active | no-permission | topic-not-exist

O status "no-permission" indica que Neptune não tem mais permissão para fazer publicações no tópico do SNS. O status "topic-not-exist" indica que o tópico foi excluído após a assinatura ser criada.

- SubscriptionCreationTime: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  A hora em que a assinatura de notificações de eventos foi criada.

`EventSubscription` é usado como o elemento de resposta para:

- CreateEventSubscription
- ModifyEventSubscription
- AddSourceIdentifierToSubscription
- RemoveSourceIdentifierFromSubscription
- DeleteEventSubscription

Outras APIs do Neptune

Ações:

- AddTagsToResource (ação)
- ListTagsForResource (ação)
- RemoveTagsFromResource (ação)
- ApplyPendingMaintenanceAction (ação)
- DescribePendingMaintenanceActions (ação)
- DescribeDBEngineVersions (ação)

Estruturas:

- DBEngineVersion (estrutura)
- EngineDefaults (estrutura)
- PendingMaintenanceAction (estrutura)
- ResourcePendingMaintenanceActions (estrutura)
- UpgradeTarget (estrutura)

- Tag (estrutura)

AddTagsToResource (ação)

O nome da CLI da AWS para essa API é: `add-tags-to-resource`.

Adiciona tags de metadados a um recurso do Amazon Neptune. Essas tags também podem ser usadas com os relatórios de alocação de custos para rastrear os custos associados a recursos do Amazon Neptune ou usadas em uma instrução de condição de uma política do IAM para o Amazon Neptune.

Solicitação

- ResourceName (na CLI: `--resource-name`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O recurso do Amazon Neptune ao qual as tags são adicionadas. Esse valor é um nome de recurso da Amazon (ARN). Para obter informações sobre a criação de um ARN, consulte Criação de um nome de recurso da Amazon (ARN) do Amazon ARN.
- Tags (na CLI: `--tags`): obrigatório: uma matriz de objetos Tag.

  As tags a serem atribuídas ao recurso do Amazon Neptune.

Resposta

- Nenhum parâmetro de resposta.

Erros

- DBInstanceNotFoundFault
- DBSnapshotNotFoundFault
- DBClusterNotFoundFault

ListTagsForResource (ação)

O nome da CLI da AWS para essa API é: `list-tags-for-resource`.

Lista todas as tags em um recurso do Amazon Neptune.

Solicitação

- Filters (na CLI: `--filters`): uma matriz de objetos Filtro.

  Não há suporte atualmente para esse parâmetro.

- ResourceName (na CLI: `--resource-name`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O recurso do Amazon Neptune com as tags a serem listadas. Esse valor é um nome de recurso da Amazon (ARN). Para obter informações sobre a criação de um ARN, consulte Criação de um nome de recurso da Amazon (ARN) do Amazon ARN.

Resposta

- TagList – Uma matriz de objetos Tag.

  Lista de tags retornadas pela operação ListTagsForResource.

Erros

- DBInstanceNotFoundFault
- DBSnapshotNotFoundFault
- DBClusterNotFoundFault

RemoveTagsFromResource (ação)

O nome da CLI da AWS para essa API é: `remove-tags-from-resource`.

Remove as tags de metadados de um recurso do Amazon Neptune.

Solicitação

- ResourceName (na CLI: `--resource-name`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O recurso do Amazon Neptune do qual as tags são removidas. Esse valor é um nome de recurso da Amazon (ARN). Para obter informações sobre a criação de um ARN, consulte Criação de um nome de recurso da Amazon (ARN) do Amazon ARN.

- TagKeys (na CLI: `--tag-keys`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  A chave (nome) da tag a ser removida.

Resposta

- Nenhum parâmetro de resposta.

Erros

- DBInstanceNotFoundFault
- DBSnapshotNotFoundFault
- DBClusterNotFoundFault

ApplyPendingMaintenanceAction (ação)

O nome da CLI da AWS para essa API é: `apply-pending-maintenance-action`.

Aplica uma ação de manutenção pendente a um recurso (por exemplo, a uma instância de banco de dados).

Solicitação

- ApplyAction (na CLI: `--apply-action`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  A ação de manutenção pendente a ser aplicada a esse recurso.

  Valores válidos: `system-update`, `db-upgrade`
- OptInType (na CLI: `--opt-in-type`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um valor que especifica o tipo de solicitação de inclusão ou desfaz uma solicitação de inclusão. Uma solicitação de inclusão do tipo `immediate` não pode ser desfeita.

  Valores válidos:

  - `immediate` – aplique a ação de manutenção imediatamente.

  - `next-maintenance` – aplique a ação de manutenção durante a próxima janela de manutenção do recurso.
  - `undo-opt-in` – Cancela todas as solicitações de inclusão `next-maintenance` existentes.
- ResourceIdentifier (na CLI: `--resource-identifier`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de recurso da Amazon (ARN) do recurso ao qual a ação de manutenção pendente se aplica. Para obter informações sobre a criação de um ARN, consulte Criação de um nome de recurso da Amazon (ARN) do Amazon ARN.

Resposta

Descreve as ações de manutenção pendentes para um recurso.

- PendingMaintenanceActionDetails – Uma matriz de objetos PendingMaintenanceAction.

  Uma lista que fornece detalhes sobre as ações de manutenção pendentes para o recurso.
- ResourceIdentifier: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN do recurso que tem ações de manutenção pendentes.

Erros

- ResourceNotFoundFault

DescribePendingMaintenanceActions (ação)

O nome da CLI da AWS para essa API é: `describe-pending-maintenance-actions`.

Retorna uma lista de recursos (por exemplo, instâncias de banco de dados) que têm pelo menos uma ação de manutenção pendente.

Solicitação

- Filters (na CLI: `--filters`): uma matriz de objetos Filtro.

  Um filtro que especifica um ou mais recursos para os quais retornar ações de manutenção pendentes.

Filtros com suporte:

- `db-cluster-id` – aceita identificadores de cluster de banco de dados e os Nomes de recursos da Amazon (ARNs) do cluster de banco de dados. A lista de recursos só incluirá ações de manutenção pendentes para os clusters de bancos de dados identificados por esses ARNs.
- `db-instance-id` – aceita identificadores de instância de banco de dados e ARNs de instâncias de banco de dados. A lista de recursos só incluirá ações de manutenção pendentes para as instâncias de bancos de dados identificadas por esses ARNs.

- Marker (na CLI: `--marker`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um token de paginação opcional fornecido por uma solicitação `DescribePendingMaintenanceActions` anterior. Se esse parâmetro for especificado, a resposta incluirá apenas os registros além do marcador, até um número de registros especificado por `MaxRecords`.
- MaxRecords (na CLI: `--max-records`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número máximo de registros a serem incluídos na resposta. Se existirem mais registros do que o valor `MaxRecords` especificado, um token de paginação chamado de marcador será incluído na resposta para que os resultados restantes possam ser recuperados.

  Padrão: 100

  Restrições: mínimo 20, máximo 100.
- ResourceIdentifier (na CLI: `--resource-identifier`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN de um recurso para o qual retornar ações de manutenção pendentes.

Resposta

- Marker: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um token de paginação opcional fornecido por uma solicitação `DescribePendingMaintenanceActions` anterior. Se esse parâmetro for especificado, a resposta incluirá apenas os registros além do marcador, até um número de registros especificado por `MaxRecords`.
- PendingMaintenanceActions – Uma matriz de objetos ResourcePendingMaintenanceActions.

Uma lista das ações de manutenção pendentes para o recurso.

Erros

- ResourceNotFoundFault

DescribeDBEngineVersions (ação)

O nome da CLI da AWS para essa API é: `describe-db-engine-versions`.

Retorna uma lista dos mecanismos de banco de dados disponíveis.

Solicitação

- DBParameterGroupFamily (na CLI: `--db-parameter-group-family`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de uma família de grupos de parâmetros de banco de dados específica para a qual retornar detalhes.

  Restrições:

  - Se fornecido, deverá ser uma DBParameterGroupFamily existente.
- DefaultOnly (na CLI: `--default-only`): um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica que somente a versão padrão do mecanismo especificado ou a combinação de mecanismo e versão principal é retornada.
- Engine (na CLI: `--engine`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O mecanismo de banco de dados a ser retornado.
- EngineVersion (na CLI: `--engine-version`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  A versão do mecanismo de banco de dados a ser retornado.

  Exemplo: `5.1.49`
- Filters (na CLI: `--filters`): uma matriz de objetos Filtro.

Sem suporte no momento.

- ListSupportedCharacterSets (na CLI: `--list-supported-character-sets`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se esse parâmetro for especificado e o mecanismo solicitado for compatível com o parâmetro `CharacterSetName` para `CreateDBInstance`, a resposta incluirá uma lista de conjuntos de caracteres com suporte para cada versão do mecanismo.

- ListSupportedTimezones (na CLI: `--list-supported-timezones`): um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se esse parâmetro for especificado e o mecanismo solicitado for compatível com o parâmetro `TimeZone` para `CreateDBInstance`, a resposta incluirá uma lista de fusos horários com suporte para cada versão do mecanismo.

- Marker (na CLI: `--marker`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um token de paginação opcional fornecido por uma solicitação anterior. Se esse parâmetro for especificado, a resposta incluirá apenas os registros além do marcador, até o valor especificado por `MaxRecords`.

- MaxRecords (na CLI: `--max-records`): um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número máximo de registros a serem incluídos na resposta. Se mais do que o valor `MaxRecords` estiver disponível, um token de paginação chamado de marcador será incluído na resposta para que os resultados a seguir possam ser recuperados.

  Padrão: 100

  Restrições: mínimo 20, máximo 100.

Resposta

- DBEngineVersions – Uma matriz de objetos DBEngineVersion.

  Uma lista de elementos `DBEngineVersion`.

- Marker: uma string, do tipo: `string` (uma string codificada em UTF-8).

Um token de paginação opcional fornecido por uma solicitação anterior. Se esse parâmetro for especificado, a resposta incluirá apenas os registros além do marcador, até o valor especificado por `MaxRecords`.

Estruturas:

DBEngineVersion (estrutura)

Esse tipo de dados é usado como um elemento de resposta na ação the section called "DescribeDBEngineVersions".

Campos

- DBEngineDescription: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  A descrição do mecanismo de banco de dados.
- DBEngineVersionDescription: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  A descrição da versão do mecanismo de banco de dados.
- DBParameterGroupFamily: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome da família de parameter groups de banco de dados para o mecanismo de banco de dados.
- Engine: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do mecanismo de banco de dados.
- EngineVersion: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O número da versão do mecanismo de banco de dados.
- ExportableLogTypes: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Os tipos de logs que o mecanismo de banco de dados tem disponíveis para exportar para o CloudWatch Logs.
- SupportedTimezones: é uma matriz de objetos Fuso horário.

  Uma lista dos fusos horários compatíveis com esse mecanismo para o parâmetro `Timezone` da ação `CreateDBInstance`.

- SupportsGlobalDatabases: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Um valor que indica se você pode usar os bancos de dados globais do Aurora com uma versão específica do mecanismo de banco de dados.
- SupportsLogExportsToCloudwatchLogs: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Um valor que indica se a versão do mecanismo oferece suporte à exportação dos tipos de log especificados pelo ExportableLogTypes para o CloudWatch Logs.
- SupportsReadReplica: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica se a versão do mecanismo de banco de dados oferece suporte a réplicas de leitura.
- ValidUpgradeTarget: é uma matriz de objetos UpgradeTarget.

  Uma lista de versões do mecanismo para as quais essa versão do mecanismo de banco de dados pode ser atualizada.

EngineDefaults (estrutura)

Contém o resultado de uma invocação bem-sucedida da ação the section called "DescribeEngineDefaultParameters".

Campos

- DBParameterGroupFamily: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o nome da família de grupos de parâmetros de banco de dados à qual os parâmetros padrão do mecanismo se aplicam.
- Marker: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um token de paginação opcional fornecido por uma solicitação EngineDefaults anterior. Se esse parâmetro for especificado, a resposta incluirá apenas os registros além do marcador, até o valor especificado por `MaxRecords`.
- Parameters: é uma matriz de objetos Parâmetro.

  Contém uma lista de parâmetros padrão do mecanismo.

`EngineDefaults` é usado como o elemento de resposta para:

- DescribeEngineDefaultParameters
- DescribeEngineDefaultClusterParameters

PendingMaintenanceAction (estrutura)

Fornece informações sobre uma ação de manutenção pendente para um recurso.

Campos

- Action: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo de ação de manutenção pendente que está disponível para o recurso.
- AutoAppliedAfterDate: é um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  A data da janela de manutenção em que a ação é aplicada. A ação de manutenção é aplicada ao recurso durante a primeira janela de manutenção após essa data. Se essa data for especificada, todas as solicitações de inclusão `next-maintenance` serão ignoradas.
- CurrentApplyDate: é um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  A data de início de vigência quando a ação de manutenção pendente é aplicada ao recurso. Essa data leva em conta as solicitações recebidas de inclusão da API the section called "ApplyPendingMaintenanceAction", `AutoAppliedAfterDate` e `ForcedApplyDate`. Esse valor estará em branco se uma solicitação de inclusão não foi recebida e nada tiver sido especificado como `AutoAppliedAfterDate` ou `ForcedApplyDate`.
- Description: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma descrição que fornece mais detalhes sobre a ação de manutenção.
- ForcedApplyDate: é um TStamp, do tipo: `timestamp` (um ponto no tempo, geralmente definido como um desvio da meia-noite de 01/01/1970).

  A data em que a ação de manutenção é aplicada automaticamente. A ação de manutenção é aplicada ao recurso nessa data, independentemente da janela de manutenção para o recurso. Se essa data for especificada, todas as solicitações de inclusão `immediate` serão ignoradas.
- OptInStatus: é uma string, do tipo: `string` (uma string codificada em UTF-8).

Indica o tipo de solicitação de inclusão que foi recebida para o recurso.

ResourcePendingMaintenanceActions (estrutura)

Descreve as ações de manutenção pendentes para um recurso.

Campos

- PendingMaintenanceActionDetails: é uma matriz de objetos PendingMaintenanceAction.

  Uma lista que fornece detalhes sobre as ações de manutenção pendentes para o recurso.
- ResourceIdentifier: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN do recurso que tem ações de manutenção pendentes.

`ResourcePendingMaintenanceActions` é usado como o elemento de resposta para:

- ApplyPendingMaintenanceAction

UpgradeTarget (estrutura)

A versão do mecanismo de banco de dados para a qual uma instância de banco de dados pode ser atualizada.

Campos

- AutoUpgrade: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Um valor que indica se a versão de destino é aplicada a todas as instâncias de banco de dados de origem que têm AutoMinorVersionUpgrade definido como true.
- Description: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  A versão do mecanismo de banco de dados para a qual uma instância de banco de dados pode ser atualizada.
- Engine: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do mecanismo de banco de dados de destino de atualização.
- EngineVersion: é uma string, do tipo: `string` (uma string codificada em UTF-8).

O número da versão do mecanismo de banco de dados de destino de atualização.

- IsMajorVersionUpgrade: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Um valor que indica se um mecanismo de banco de dados é atualizado para uma versão principal.

- SupportsGlobalDatabases: é um BooleanOptional, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Um valor que indica se você pode usar os bancos de dados globais do Neptune com a versão do mecanismo de destino.

Tag (estrutura)

Metadados atribuídos a um recurso do Amazon Neptune consistente de um par chave/valor.

Campos

- Key: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma chave é o nome obrigatório da tag. O valor da string pode ter de 1 a 128 caracteres Unicode e não pode ter os prefixos `aws:` ou `rds:`. A string pode conter apenas o conjunto de letras em Unicode, dígitos, espaço em branco, “_”, “.”, “/”, “=”, “+”, “-” (Java regex: “^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$”).

- Value: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um valor é o valor opcional da tag. O valor da string pode ter de 1 a 256 caracteres Unicode e não pode ter os prefixos `aws:` ou `rds:`. A string pode conter apenas o conjunto de letras em Unicode, dígitos, espaço em branco, “_”, “.”, “/”, “=”, “+”, “-” (Java regex: “^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$”).

Tipos de dados comuns do Neptune

Estruturas:

- AvailabilityZone (estrutura)
- DBSecurityGroupMembership (estrutura)
- DomainMembership (estrutura)

- DoubleRange (estrutura)
- Endpoint (estrutura)
- Filtro (estrutura)
- Intervalo (estrutura)
- ServerlessV2ScalingConfiguration (estrutura)
- ServerlessV2ScalingConfigurationInfo (estrutura)
- Fuso horário (estrutura)
- VpcSecurityGroupMembership (estrutura)

AvailabilityZone (estrutura)

Especifica uma zona de disponibilidade.

Campos

- Name: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome da zona de disponibilidade.

DBSecurityGroupMembership (estrutura)

Especifica a associação em um grupo de segurança de banco de dados designado.

Campos

- DBSecurityGroupName: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do grupo de segurança de banco de dados.
- Status: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O status do grupo de segurança de banco de dados.

DomainMembership (estrutura)

Um registro de associação do domínio do Active Directory associado a uma instância de banco de dados.

Campos

- Domain: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador do domínio do Active Directory.
- FQDN: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de domínio totalmente qualificado do domínio do Active Directory.
- IAMRoleName: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome da função do IAM a ser usada ao fazer chamadas de API para o Directory Service.
- Status: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O status da associação do domínio do Active Directory da instância de banco de dados, por exemplo, associada, associação pendente, com falhou, etc.

DoubleRange (estrutura)

Um intervalo de valores duplos.

Campos

- From: é um Double, do tipo: `double` (um número de ponto flutuante IEEE 754 de dupla precisão).

  O valor mínimo do intervalo.
- To: é um Double, do tipo: `double` (um número de ponto flutuante IEEE 754 de dupla precisão).

  O valor máximo do intervalo.

Endpoint (estrutura)

Especifica um endpoint de conexão.

Para a estrutura de dados que representa endpoints de cluster de banco de dados do Amazon Neptune, consulte `DBClusterEndpoint`.

Campos

- Address: é uma string, do tipo: `string` (uma string codificada em UTF-8).

Especifica o endereço DNS da instância de banco de dados.

- HostedZoneId: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Especifica o ID que o Amazon Route 53 atribui ao criar uma zona hospedada.

- Port: é um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Especifica a porta em que o mecanismo de banco de dados está realizando a escuta.

Filtro (estrutura)

No momento, não há suporte para esse tipo.

Campos

- Name: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Não há suporte atualmente para esse parâmetro.

- Values: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Não há suporte atualmente para esse parâmetro.

Intervalo (estrutura)

Um intervalo de valores inteiros.

Campos

- From: é um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O valor mínimo do intervalo.

- Step: é um IntegerOptional, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O valor da etapa para o intervalo. Por exemplo, se você tiver um intervalo de 5.000 a 10.000, com um valor de etapa de 1.000, os valores válidos começarão em 5.000 e irão até 1.000. Embora 7.500 esteja dentro do intervalo, não se trata de um valor válido para o intervalo. Os valores válidos são 5.000, 6.000, 7.000, 8.000...

- To: é um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

O valor máximo do intervalo.

ServerlessV2ScalingConfiguration (estrutura)

Contém a configuração de escalabilidade de um cluster de banco de dados do Neptune Serverless.

Para obter mais informações, consulte Using Amazon Neptune Serverless no Guia do usuário do Amazon Neptune.

Campos

- MaxCapacity: é um DoubleOptional, do tipo: `double` (um número de ponto flutuante IEEE 754 de dupla precisão).

  O número máximo de unidades de capacidade do Neptune (NCUs) para uma instância de banco de dados em um cluster do Neptune Serverless. É possível especificar valores de NCU em incrementos de meio passo, por exemplo, 40, 40,5, 41, etc.
- MinCapacity: é um DoubleOptional, do tipo: `double` (um número de ponto flutuante IEEE 754 de dupla precisão).

  O número mínimo de unidades de capacidade do Neptune (NCUs) para uma instância de banco de dados em um cluster do Neptune Serverless. É possível especificar valores de NCU em incrementos de meio passo, por exemplo, 8, 8,5, 9, etc.

ServerlessV2ScalingConfigurationInfo (estrutura)

Mostra a configuração de escalabilidade de um cluster de banco de dados do Neptune Serverless.

Para obter mais informações, consulte Using Amazon Neptune Serverless no Guia do usuário do Amazon Neptune.

Campos

- MaxCapacity: é um DoubleOptional, do tipo: `double` (um número de ponto flutuante IEEE 754 de dupla precisão).

  O número máximo de unidades de capacidade do Neptune (NCUs) para uma instância de banco de dados em um cluster do Neptune Serverless. É possível especificar valores de NCU em incrementos de meio passo, por exemplo, 40, 40,5, 41, etc.

- MinCapacity: é um DoubleOptional, do tipo: `double` (um número de ponto flutuante IEEE 754 de dupla precisão).

  O número mínimo de unidades de capacidade do Neptune (NCUs) para uma instância de banco de dados em um cluster do Neptune Serverless. É possível especificar valores de NCU em incrementos de meio passo, por exemplo, 8, 8,5, 9, etc.

Fuso horário (estrutura)

Um fuso horário associado a um domínio the section called “DBInstance”.

Campos

- TimezoneName: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do fuso horário.

VpcSecurityGroupMembership (estrutura)

Esse tipo de dados é usado como um elemento de resposta para consultas na associação do grupo de segurança da VPC.

Campos

- Status: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O status do grupo de segurança da VPC.
- VpcSecurityGroupId: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do grupo de segurança da VPC.

Exceções do Neptune específicas de APIs individuais

Exceções:

- AuthorizationAlreadyExistsFault (estrutura)
- AuthorizationNotFoundFault (estrutura)
- AuthorizationQuotaExceededFault (estrutura)

- CertificateNotFoundFault (estrutura)
- DBClusterAlreadyExistsFault (estrutura)
- DBClusterNotFoundFault (estrutura)
- DBClusterParameterGroupNotFoundFault (estrutura)
- DBClusterQuotaExceededFault (estrutura)
- DBClusterRoleAlreadyExistsFault (estrutura)
- DBClusterRoleNotFoundFault (estrutura)
- DBClusterRoleQuotaExceededFault (estrutura)
- DBClusterSnapshotAlreadyExistsFault (estrutura)
- DBClusterSnapshotNotFoundFault (estrutura)
- DBInstanceAlreadyExistsFault (estrutura)
- DBInstanceNotFoundFault (estrutura)
- DBLogFileNotFoundFault (estrutura)
- DBParameterGroupAlreadyExistsFault (estrutura)
- DBParameterGroupNotFoundFault (estrutura)
- DBParameterGroupQuotaExceededFault (estrutura)
- DBSecurityGroupAlreadyExistsFault (estrutura)
- DBSecurityGroupNotFoundFault (estrutura)
- DBSecurityGroupNotSupportedFault (estrutura)
- DBSecurityGroupQuotaExceededFault (estrutura)
- DBSnapshotAlreadyExistsFault (estrutura)
- DBSnapshotNotFoundFault (estrutura)
- DBSubnetGroupAlreadyExistsFault (estrutura)
- DBSubnetGroupDoesNotCoverEnoughAZs (estrutura)
- DBSubnetGroupNotAllowedFault (estrutura)
- DBSubnetGroupNotFoundFault (estrutura)
- DBSubnetGroupQuotaExceededFault (estrutura)
- DBSubnetQuotaExceededFault (estrutura)
- DBUpgradeDependencyFailureFault (estrutura)

- DomainNotFoundFault (estrutura)
- EventSubscriptionQuotaExceededFault (estrutura)
- GlobalClusterAlreadyExistsFault (estrutura)
- GlobalClusterNotFoundFault (estrutura)
- GlobalClusterQuotaExceededFault (estrutura)
- InstanceQuotaExceededFault (estrutura)
- InsufficientDBClusterCapacityFault (estrutura)
- InsufficientDBInstanceCapacityFault (estrutura)
- InsufficientStorageClusterCapacityFault (estrutura)
- InvalidDBClusterEndpointStateFault (estrutura)
- InvalidDBClusterSnapshotStateFault (estrutura)
- InvalidDBClusterStateFault (estrutura)
- InvalidDBInstanceStateFault (estrutura)
- InvalidDBParameterGroupStateFault (estrutura)
- InvalidDBSecurityGroupStateFault (estrutura)
- InvalidDBSnapshotStateFault (estrutura)
- InvalidDBSubnetGroupFault (estrutura)
- InvalidDBSubnetGroupStateFault (estrutura)
- InvalidDBSubnetStateFault (estrutura)
- InvalidEventSubscriptionStateFault (estrutura)
- InvalidGlobalClusterStateFault (estrutura)
- InvalidOptionGroupStateFault (estrutura)
- InvalidRestoreFault (estrutura)
- InvalidSubnet (estrutura)
- InvalidVPCNetworkStateFault (estrutura)
- KMSKeyNotAccessibleFault (estrutura)
- OptionGroupNotFoundFault (estrutura)
- PointInTimeRestoreNotEnabledFault (estrutura)
- ProvisionedIopsNotAvailableInAZFault (estrutura)
- ResourceNotFoundFault (estrutura)

- SNSInvalidTopicFault (estrutura)
- SNSNoAuthorizationFault (estrutura)
- SNSTopicArnNotFoundFault (estrutura)
- SharedSnapshotQuotaExceededFault (estrutura)
- SnapshotQuotaExceededFault (estrutura)
- SourceNotFoundFault (estrutura)
- StorageQuotaExceededFault (estrutura)
- StorageTypeNotSupportedFault (estrutura)
- SubnetAlreadyInUse (estrutura)
- SubscriptionAlreadyExistFault (estrutura)
- SubscriptionCategoryNotFoundFault (estrutura)
- SubscriptionNotFoundFault (estrutura)

AuthorizationAlreadyExistsFault (estrutura)

Código de status HTTP retornado: 400.

O grupo de segurança do EC2 ou CIDRIP especificado já está autorizado para o grupo de segurança de banco de dados.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

AuthorizationNotFoundFault (estrutura)

Código de status HTTP retornado: 404.

O grupo de segurança do EC2 ou CIDRIP especificado não está autorizado para o grupo de segurança de banco de dados determinado.

O Neptune também não pode estar autorizado por meio do IAM para executar as ações necessárias em seu nome.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

AuthorizationQuotaExceededFault (estrutura)

Código de status HTTP retornado: 400.

A cota de autorização de grupo de segurança de banco de dados foi atingida.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

CertificateNotFoundFault (estrutura)

Código de status HTTP retornado: 404.

CertificateIdentifier não se refere a um certificado existente.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

DBClusterAlreadyExistsFault (estrutura)

Código de status HTTP retornado: 400.

O usuário já tem um cluster de banco de dados com o identificador determinado.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

DBClusterNotFoundFault (estrutura)

Código de status HTTP retornado: 404.

DBClusterIdentifier não se refere a um cluster de banco de dados existente.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

DBClusterParameterGroupNotFoundFault (estrutura)

Código de status HTTP retornado: 404.

DBClusterParameterGroupName não se refere a um grupo de parâmetros de cluster de banco de dados existente.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

DBClusterQuotaExceededFault (estrutura)

Código de status HTTP retornado: 403.

O usuário tentou criar um novo cluster de banco de dados e o usuário já atingiu a cota máxima permitido de cluster de banco de dados.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

DBClusterRoleAlreadyExistsFault (estrutura)

Código de status HTTP retornado: 400.

O nome de recurso da Amazon (ARN) da função do IAM especificada já está associado ao cluster de banco de dados determinado.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

DBClusterRoleNotFoundFault (estrutura)

Código de status HTTP retornado: 404.

O nome de recurso da Amazon (ARN) da função do IAM especificada não está associado ao cluster de banco de dados determinado.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

DBClusterRoleQuotaExceededFault (estrutura)

Código de status HTTP retornado: 400.

Você excedeu o número máximo de funções do IAM que podem ser associadas ao cluster de banco de dados especificado.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

DBClusterSnapshotAlreadyExistsFault (estrutura)

Código de status HTTP retornado: 400.

O usuário já tem um snapshot de cluster de banco de dados com o identificador determinado.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

DBClusterSnapshotNotFoundFault (estrutura)

Código de status HTTP retornado: 404.

DBClusterSnapshotIdentifier não se refere a um snapshot de cluster de banco de dados existente.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

DBInstanceAlreadyExistsFault (estrutura)

Código de status HTTP retornado: 400.

O usuário já tem uma instância de banco de dados com o identificador determinado.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

DBInstanceNotFoundFault (estrutura)

Código de status HTTP retornado: 404.

DBInstanceIdentifier não se refere a uma instância de banco de dados existente.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

DBLogFileNotFoundFault (estrutura)

Código de status HTTP retornado: 404.

LogFileName não se refere a um arquivo de log de banco de dados existente.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

DBParameterGroupAlreadyExistsFault (estrutura)

Código de status HTTP retornado: 400.

Já existe um grupo de parâmetros de banco de dados com o mesmo nome.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

DBParameterGroupNotFoundFault (estrutura)

Código de status HTTP retornado: 404.

DBParameterGroupName não se refere a um grupo de parâmetros de banco de dados existente.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

DBParameterGroupQuotaExceededFault (estrutura)

Código de status HTTP retornado: 400.

A solicitação faria com que o usuário excedesse o número permitido de grupos de parâmetro de banco de dados.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

DBSecurityGroupAlreadyExistsFault (estrutura)

Código de status HTTP retornado: 400.

Já existe um grupo de segurança de banco de dados com o nome especificado em DBSecurityGroupName.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

DBSecurityGroupNotFoundFault (estrutura)

Código de status HTTP retornado: 404.

DBSecurityGroupName não se refere a um grupo de segurança de banco de dados existente.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

DBSecurityGroupNotSupportedFault (estrutura)

Código de status HTTP retornado: 400.

Um grupo de segurança de banco de dados não é permitido para essa ação.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

DBSecurityGroupQuotaExceededFault (estrutura)

Código de status HTTP retornado: 400.

A solicitação faria com que o usuário excedesse o número permitido de grupos de segurança de banco de dados.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

DBSnapshotAlreadyExistsFault (estrutura)

Código de status HTTP retornado: 400.

DBSnapshotIdentifier já é usado por um snapshot existente.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

DBSnapshotNotFoundFault (estrutura)

Código de status HTTP retornado: 404.

DBSnapshotIdentifier não se refere a um DB snapshot existente.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

Uma mensagem descrevendo os detalhes do problema.

DBSubnetGroupAlreadyExistsFault (estrutura)

Código de status HTTP retornado: 400.

DBSubnetGroupName já é usado por um grupo de sub-redes de banco de dados.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

DBSubnetGroupDoesNotCoverEnoughAZs (estrutura)

Código de status HTTP retornado: 400.

As sub-redes no grupo de sub-redes de banco de dados devem abranger pelo menos duas zonas de disponibilidade, a menos que haja apenas uma zona de disponibilidade.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

DBSubnetGroupNotAllowedFault (estrutura)

Código de status HTTP retornado: 400.

Indica que o DBSubnetGroup não deve ser especificado ao criar réplicas de leitura que estão na mesma região que a instância de origem.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

DBSubnetGroupNotFoundFault (estrutura)

Código de status HTTP retornado: 404.

DBSubnetGroupName não se refere a um grupo de sub-redes de banco de dados existente.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

DBSubnetGroupQuotaExceededFault (estrutura)

Código de status HTTP retornado: 400.

A solicitação faria com que o usuário excedesse o número permitido de grupos de sub-redes de banco de dados.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

DBSubnetQuotaExceededFault (estrutura)

Código de status HTTP retornado: 400.

A solicitação faria com que o usuário excedesse o número permitido de sub-redes em um grupos de sub-redes de banco de dados.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

DBUpgradeDependencyFailureFault (estrutura)

Código de status HTTP retornado: 400.

Ocorreu uma falha na atualização do banco de dados porque um recurso do qual depende o banco de dados não pôde ser modificado.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

DomainNotFoundFault (estrutura)

Código de status HTTP retornado: 404.

Domain não se refere a um domínio existente do Active Directory.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

EventSubscriptionQuotaExceededFault (estrutura)

Código de status HTTP retornado: 400.

Você excedeu o número de eventos que pode assinar.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

GlobalClusterAlreadyExistsFault (estrutura)

Código de status HTTP retornado: 400.

O `GlobalClusterIdentifier` já existe. Escolha um novo identificador de banco de dados global (nome exclusivo) para criar um cluster de banco de dados global.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

GlobalClusterNotFoundFault (estrutura)

Código de status HTTP retornado: 404.

O `GlobalClusterIdentifier` não se refere a um cluster de banco de dados global existente.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

GlobalClusterQuotaExceededFault (estrutura)

Código de status HTTP retornado: 400.

O número de clusters de banco de dados global para essa conta já está no máximo permitido.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

InstanceQuotaExceededFault (estrutura)

Código de status HTTP retornado: 400.

A solicitação faria com que o usuário excedesse o número permitido de instâncias de banco de dados.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

Uma mensagem descrevendo os detalhes do problema.

InsufficientDBClusterCapacityFault (estrutura)

Código de status HTTP retornado: 403.

O cluster de banco de dados não tem capacidade suficiente para a operação atual.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

InsufficientDBInstanceCapacityFault (estrutura)

Código de status HTTP retornado: 400.

A classe de instância de banco de dados especificada não está disponível na zona de disponibilidade determinada.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

InsufficientStorageClusterCapacityFault (estrutura)

Código de status HTTP retornado: 400.

Não há armazenamento suficiente disponível para a ação atual. Você pode resolver esse erro atualizando seu grupo de sub-redes para usar outras zonas de disponibilidade que tenham mais espaço de armazenamento disponível.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

InvalidDBClusterEndpointStateFault (estrutura)

Código de status HTTP retornado: 400.

A operação solicitada não poderá ser executada no endpoint enquanto o endpoint estiver nesse estado.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

InvalidDBClusterSnapshotStateFault (estrutura)

Código de status HTTP retornado: 400.

O valor fornecido não é um estado de snapshot de cluster de banco de dados válido.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

InvalidDBClusterStateFault (estrutura)

Código de status HTTP retornado: 400.

O cluster de banco de dados não está em um estado válido.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

InvalidDBInstanceStateFault (estrutura)

Código de status HTTP retornado: 400.

A instância de banco de dados especificada não está no estado disponível.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

InvalidDBParameterGroupStateFault (estrutura)

Código de status HTTP retornado: 400.

O grupo de parâmetros de banco de dados está em uso ou está em um estado inválido. Se você estiver tentando excluir o grupo de parâmetros, não poderá excluí-lo quando o grupo de parâmetros estiver nesse estado.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

InvalidDBSecurityGroupStateFault (estrutura)

Código de status HTTP retornado: 400.

O estado do grupo de segurança de banco de dados não permite a exclusão.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

InvalidDBSnapshotStateFault (estrutura)

Código de status HTTP retornado: 400.

O estado do DB snapshot não permite a exclusão.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

InvalidDBSubnetGroupFault (estrutura)

Código de status HTTP retornado: 400.

Indica que o DBSubnetGroup não pertence à mesma VPC que a de uma réplica de leitura existente em várias regiões da mesma instância de origem.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

InvalidDBSubnetGroupStateFault (estrutura)

Código de status HTTP retornado: 400.

O grupo de sub-redes de banco de dados não pode ser excluído porque está em uso.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

InvalidDBSubnetStateFault (estrutura)

Código de status HTTP retornado: 400.

A sub-rede de banco de dados especificada não está no estado disponível.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

Uma mensagem descrevendo os detalhes do problema.

InvalidEventSubscriptionStateFault (estrutura)

Código de status HTTP retornado: 400.

A assinatura de evento está em um estado inválido.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

InvalidGlobalClusterStateFault (estrutura)

Código de status HTTP retornado: 400.

O cluster global está em um estado inválido e não pode realizar a operação solicitada.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

InvalidOptionGroupStateFault (estrutura)

Código de status HTTP retornado: 400.

O grupo de opções não está no estado disponível.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

InvalidRestoreFault (estrutura)

Código de status HTTP retornado: 400.

Não é possível fazer a restauração do backup de vpc para uma instância de banco de dados que não seja da vpc.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

InvalidSubnet (estrutura)

Código de status HTTP retornado: 400.

A sub-rede solicitada é inválida ou foram solicitadas várias sub-redes que não estão em uma VPC comum.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

InvalidVPCNetworkStateFault (estrutura)

Código de status HTTP retornado: 400.

O grupo de sub-redes de banco de dados não abrange todas as zonas de disponibilidade depois que ele é criado devido à alteração dos usuários.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

KMSKeyNotAccessibleFault (estrutura)

Código de status HTTP retornado: 400.

Erro ao acessar a chave do KMS.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

OptionGroupNotFoundFault (estrutura)

Código de status HTTP retornado: 404.

O grupo de opções designado não foi encontrado.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

PointInTimeRestoreNotEnabledFault (estrutura)

Código de status HTTP retornado: 400.

SourceDBInstanceIdentifier refere-se a uma instância de banco de dados com BackupRetentionPeriod igual a 0.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

ProvisionedIopsNotAvailableInAZFault (estrutura)

Código de status HTTP retornado: 400.

IOPS provisionadas não disponíveis na zona de disponibilidade especificada.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

ResourceNotFoundFault (estrutura)

Código de status HTTP retornado: 404.

O ID do recurso especificado não foi encontrado.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

SNSInvalidTopicFault (estrutura)

Código de status HTTP retornado: 400.

O tópico do SNS é inválido.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

SNSNoAuthorizationFault (estrutura)

Código de status HTTP retornado: 400.

Não há autorização do SNS.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

Uma mensagem descrevendo os detalhes do problema.

SNSTopicArnNotFoundFault (estrutura)

Código de status HTTP retornado: 404.

O ARN do tópico do SNS não foi encontrado.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

SharedSnapshotQuotaExceededFault (estrutura)

Código de status HTTP retornado: 400.

Você excedeu o número máximo de contas com as quais você pode compartilhar um DB snapshot manual.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

SnapshotQuotaExceededFault (estrutura)

Código de status HTTP retornado: 400.

A solicitação faria com que o usuário excedesse o número permitido de DB snapshots.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

SourceNotFoundFault (estrutura)

Código de status HTTP retornado: 404.

A origem não foi encontrada.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

StorageQuotaExceededFault (estrutura)

Código de status HTTP retornado: 400.

A solicitação faria com que o usuário excedesse o volume de armazenamento disponível em todas as instâncias de banco de dados.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

StorageTypeNotSupportedFault (estrutura)

Código de status HTTP retornado: 400.

StorageType especificado não pode ser associado à instância de banco de dados.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

SubnetAlreadyInUse (estrutura)

Código de status HTTP retornado: 400.

A sub-rede de banco de dados já está em uso na zona de disponibilidade.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

SubscriptionAlreadyExistFault (estrutura)

Código de status HTTP retornado: 400.

Essa assinatura já existe.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

SubscriptionCategoryNotFoundFault (estrutura)

Código de status HTTP retornado: 404.

A categoria de assinatura designada não foi encontrada.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem descrevendo os detalhes do problema.

SubscriptionNotFoundFault (estrutura)

Código de status HTTP retornado: 404.

A assinatura designada não foi encontrada.

Campos

- message: é uma ExceptionMessage, do tipo: `string` (uma string codificada em UTF-8).

Uma mensagem descrevendo os detalhes do problema.

Referência da API de dados do Amazon Neptune

Este capítulo documenta as operações da API de dados do Neptune que você pode usar para carregar, acessar e manter os dados no grafo do Neptune.

A API de dados do Neptune é compatível com o SDK para carregar dados, executar consultas, obter informações sobre os dados e executar operações de machine learning. Ela é compatível com as linguagens de consulta Gremlin e openCypher no Neptune e está disponível em todas as linguagens do SDK. Ela assina automaticamente as solicitações da API e simplifica consideravelmente a integração do Neptune às aplicações.

Sumário

- BulkLoadIdNotFoundException (estrutura)
- CancelledByUserException (estrutura)
- ClientTimeoutException (estrutura)
- ConcurrentModificationException (estrutura)
- ConstraintViolationException (estrutura)
- ExpiredStreamException (estrutura)
- FailureByQueryException (estrutura)
- IllegalArgumentException (estrutura)
- InternalFailureException (estrutura)
- InvalidArgumentException (estrutura)
- InvalidNumericDataException (estrutura)
- InvalidParameterException (estrutura)
- LoadUrlAccessDeniedException (estrutura)
- MalformedQueryException (estrutura)
- MemoryLimitExceededException (estrutura)
- MethodNotAllowedException (estrutura)
- MissingParameterException (estrutura)
- MLResourceNotFoundException (estrutura)
- ParsingException (estrutura)
- PreconditionsFailedException (estrutura)
- QueryLimitExceededException (estrutura)
- QueryLimitException (estrutura)
- QueryTooLargeException (estrutura)
- ReadOnlyViolationException (estrutura)
- S3Exception (estrutura)
- ServerShutdownException (estrutura)
- StatisticsNotAvailableException (estrutura)
- StreamRecordsNotFoundException (estrutura)
- ThrottlingException (estrutura)
- TimeLimitExceededException (estrutura)

- TooManyRequestsException (estrutura)
- UnsupportedOperationException (estrutura)
- UnloadUrlAccessDeniedException (estrutura)

Mecanismo de plano de dados do Neptune, redefinição rápida e APIs de estrutura geral

Operações do mecanismo:

- GetEngineStatus (ação)
- ExecuteFastReset (ação)

Estruturas de operação do mecanismo:

- QueryLanguageVersion (estrutura)
- FastResetToken (estrutura)

GetEngineStatus (ação)

> O nome da CLI da AWS para essa API é: `get-engine-status`.

Recupera o status do banco de dados de grafos no host.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:GetEngineStatus nesse cluster.

Solicitação

- Sem parâmetros de solicitação.

Resposta

- dbEngineVersion: uma string, do tipo: `string` (uma string codificada em UTF-8).

Defina como a versão do mecanismo do Neptune em execução no cluster de banco de dados. Se essa versão do mecanismo tiver sido corrigida manualmente desde que foi lançada, o número da versão terá o prefixo `Patch-`.

- dfeQueryEngine: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Defina como `enabled` se o mecanismo do DFE estiver totalmente habilitado, ou `viaQueryHint` (o padrão) se o mecanismo do DFE for usado somente com consultas que tenham a dica de consulta `useDFE` definida como `true`.
- features: uma matriz de mapa dos pares de valor-chave em que:

  Cada chave é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Cada valor é um documento, do tipo: `document` (um conteúdo aberto independente de protocolo representado por um modelo de dados semelhante a JSON).

  Contém informações de status sobre os atributos habilitados no cluster de banco de dados.
- gremlin: um objeto QueryLanguageVersion.

  Contém informações sobre a linguagem de consulta Gremlin disponível no cluster. Especificamente, contém um campo de versão que especifica a versão atual do TinkerPop que está sendo usada pelo mecanismo.
- labMode: uma matriz de mapa dos pares de valor-chave em que:

  Cada chave é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Cada valor é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém as configurações do modo de laboratório usadas pelo mecanismo.
- opencypher: um objeto QueryLanguageVersion.

  Contém informações sobre a linguagem de consulta openCypher disponível no cluster. Especificamente, contém um campo de versão que especifica a versão atual do openCypher que está sendo usada pelo mecanismo.
- role: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Defina como `reader` se a instância for uma réplica de leitura ou como `writer` se a instância for a principal.
- rollingBackTrxCount: um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

Se houver transações que estiverem sendo revertidas, esse campo será definido como o número dessas transações. Se não houver nenhuma, o campo não aparecerá.

- rollingBackTrxEarliestStartTime: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Defina como a hora de início da transação mais antiga revertida. Se nenhuma transação estiver sendo revertida, o campo não aparecerá.

- settings: uma matriz de mapa dos pares de valor-chave em que:

  Cada chave é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Cada valor é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém informações sobre as configurações atuais no cluster de banco de dados. Por exemplo, contém a configuração de tempo limite da consulta de cluster atual (`clusterQueryTimeoutInMs`).

- sparql: um objeto QueryLanguageVersion.

  Contém informações sobre a linguagem de consulta SPARQL disponível no cluster. Especificamente, contém um campo de versão que especifica a versão atual do SPARQL que está sendo usada pelo mecanismo.

- startTime: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Defina como a hora UTC em que o processo do servidor atual foi iniciado.

- status: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Defina como `healthy` se a instância não estiver com problemas. Se a instância estiver se recuperando de um travamento ou sendo reinicializada e houver transações ativas em execução no desligamento do servidor mais recente, o status será definido como `recovery`.

Erros

- UnsupportedOperationException
- InternalFailureException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException

- InvalidArgumentException
- IllegalArgumentException
- TooManyRequestsException

ExecuteFastReset (ação)

O nome da CLI da AWS para essa API é: `execute-fast-reset`.

A API REST de redefinição rápida permite redefinir um grafo do Neptune com rapidez e facilidade, removendo todos seus dados.

A redefinição rápida do Neptune é um processo em duas etapas. Primeiro você chama `ExecuteFastReset` com `action` definido como `initiateDatabaseReset`. Isso gera um token UUID que você inclui ao chamar `ExecuteFastReset` novamente com `action` definido como `performDatabaseReset`. Consulte Esvaziar um cluster de banco de dados do Amazon Neptune usando a API de redefinição rápida.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:ResetDatabase nesse cluster.

Solicitação

- action (na CLI :`--action`): obrigatório: uma ação, do tipo: `string` (uma string codificada em UTF-8).

  A ação de redefinição rápida. Um dos seguintes valores:

  - **`initiateDatabaseReset`**: essa ação gera um token exclusivo necessário para realmente realizar a redefinição rápida.
  - **`performDatabaseReset`**: essa ação usa o token gerado pela ação `initiateDatabaseReset` para realmente realizar a redefinição rápida.

- token (na CLI: `--token`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O token de redefinição rápida para iniciar a redefinição.

Resposta

- payload: um objeto FastResetToken.

  O `payload` só é gerado pela ação `initiateDatabaseReset` e contém o token exclusivo a ser usado com a ação `performDatabaseReset` para ocasionar a redefinição.
- status: obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O `status` é gerado somente para a ação `performDatabaseReset` e indica se a solicitação de redefinição rápida foi aceita ou não.

Erros

- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- ServerShutdownException
- PreconditionsFailedException
- MethodNotAllowedException
- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

Estruturas de operação do mecanismo:

QueryLanguageVersion (estrutura)

Estrutura para expressar a versão da linguagem de consulta.

Campos

- version: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  A versão da linguagem de consulta.

FastResetToken (estrutura)

Uma estrutura que contém o token de redefinição rápida usado para iniciar uma redefinição rápida.

Campos

- token: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um UUID gerado pelo banco de dados na ação `initiateDatabaseReset` e depois consumido por `performDatabaseReset` para redefinir o banco de dados.

APIs de consulta do Neptune

Ações de consulta do Gremlin:

- ExecuteGremlinQuery (ação)
- ExecuteGremlinExplainQuery (ação)
- ExecuteGremlinProfileQuery (ação)
- ListGremlinQueries (ação)
- GetGremlinQueryStatus (ação)
- CancelGremlinQuery (ação)

Ações de consulta do openCypher:

- ExecuteOpenCypherQuery (ação)
- ExecuteOpenCypherExplainQuery (ação)
- ListOpenCypherQueries (ação)
- GetOpenCypherQueryStatus (ação)
- CancelOpenCypherQuery (ação)

Estruturas de consulta:

- QueryEvalStats (estrutura)
- GremlinQueryStatus (estrutura)
- GremlinQueryStatusAttributes (estrutura)

ExecuteGremlinQuery (ação)

O nome da CLI da AWS para essa API é: `execute-gremlin-query`.

Esse comando executa uma consulta do Gremlin. O Amazon Neptune é compatível com o Apache TinkerPop3 e o Gremlin, então você pode usar a linguagem de percursos Gremlin para consultar o grafo, conforme descrito em The Graph na documentação do Apache TinkerPop3. Mais detalhes também podem ser encontrados em Acessar o grafo do Neptune com o Gremlin.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize uma das seguintes ações do IAM nesse cluster, dependendo da consulta:

- neptune-db:ReadDataViaQuery
- neptune-db:WriteDataViaQuery
- neptune-db:DeleteDataViaQuery

Observe que a chave de condição do IAM neptune-db:QueryLanguage:Gremlin pode ser usada no documento de política para restringir o uso de consultas do Gremlin (consulte Chaves de condição disponíveis em declarações de política de acesso a dados do IAM do Neptune).

Solicitação

- gremlinQuery (na CLI: `--gremlin-query`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Usando essa API, é possível executar consultas do Gremlin no formato de string da mesma forma que é possível no uso do endpoint HTTP. A interface é compatível com qualquer versão do Gremlin que o cluster de banco de dados esteja usando (consulte a seção sobre o cliente do Tinkerpop para determinar qual versão do Gremlin é compatível com a versão do mecanismo).
- serializer (na CLI: `--serializer`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Se não forem nulos, os resultados da consulta serão gerados em uma mensagem de resposta serializada no formato especificado por esse parâmetro. Consulte a seção GraphSON na documentação do TinkerPop para obter uma lista dos formatos compatíveis no momento.

Resposta

- meta: um documento, do tipo `document` (um conteúdo aberto independente de protocolo representado por um modelo de dados semelhante a JSON).

  Metadados sobre a consulta do Gremlin.
- requestId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador exclusivo da consulta do Gremlin.
- result: um documento, do tipo `document` (um conteúdo aberto independente de protocolo representado por um modelo de dados semelhante a JSON).

  A saída da consulta do Gremlin do servidor.
- status – Um objeto GremlinQueryStatusAttributes.

  O status da consulta do Gremlin.

Erros

- QueryTooLargeException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException
- ParsingException
- ConstraintViolationException

- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

ExecuteGremlinExplainQuery (ação)

O nome da CLI da AWS para essa API é: `execute-gremlin-explain-query`.

Executa uma consulta do Gremlin Explain.

O Amazon Neptune adicionou um atributo do Gremlin denominado `explain` que oferece uma ferramenta de autoatendimento para entender a abordagem de execução adotada pelo mecanismo do Neptune para a consulta. Você o invoca adicionando um parâmetro `explain` a uma chamada HTTP que envia uma consulta do Gremlin.

O atributo explain fornece informações sobre a estrutura lógica dos planos de execução da consulta. É possível usar essas informações para identificar possíveis gargalos de execução e de avaliação e ajustar a consulta, conforme explicado em Ajustar consultas do Gremlin. Você também pode usar as dicas de consulta para melhorar os planos de execução das consultas.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize uma das seguintes ações do IAM nesse cluster, dependendo da consulta:

- neptune-db:ReadDataViaQuery
- neptune-db:WriteDataViaQuery
- neptune-db:DeleteDataViaQuery

Observe que a chave de condição do IAM neptune-db:QueryLanguage:Gremlin pode ser usada no documento de política para restringir o uso de consultas do Gremlin (consulte Chaves de condição disponíveis em declarações de política de acesso a dados do IAM do Neptune).

Solicitação

- gremlinQuery (na CLI: `--gremlin-query`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

A string de consulta do explain do Gremlin.

Resposta

- output: um ReportAsText, do tipo: `blob` (um bloco de dados binários não interpretados).

  Um blob de texto que contém o resultado de explain do Gremlin, conforme descrito em Ajustar consultas do Gremlin.

Erros

- QueryTooLargeException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

ExecuteGremlinProfileQuery (ação)

O nome da CLI da AWS para essa API é: `execute-gremlin-profile-query`.

Executa a consulta de perfil do Gremlin, que executa um percurso especificado, coleta várias métricas sobre a execução e produz um relatório de perfil como saída. Consulte API de perfil do Gremlin no Neptune para obter detalhes.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:ReadDataViaQuery nesse cluster.

Observe que a chave de condição do IAM neptune-db:QueryLanguage:Gremlin pode ser usada no documento de política para restringir o uso de consultas do Gremlin (consulte Chaves de condição disponíveis em declarações de política de acesso a dados do IAM do Neptune).

Solicitação

- chop (na CLI: `--chop`): um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Se diferente de zero, fará com que a string de resultados seja truncada nesse número de caracteres. Se definido como zero, a string conterá todos os resultados.
- gremlinQuery (na CLI: `--gremlin-query`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  A string de consulta do Gremlin para o perfil.
- indexOps (na CLI: `--index-ops`): um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se esse sinalizador for definido como `TRUE`, os resultados incluirão um relatório detalhado de todas as operações de índice que ocorreram durante a execução e a serialização da consulta.
- results (na CLI: `--results`): um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se esse sinalizador for definido como `TRUE`, os resultados da consulta serão coletados e exibidos como parte do relatório de perfil. Se `FALSE`, somente a contagem de resultados será exibida.
- serializer (na CLI: `--serializer`): uma string, do tipo: `string` (uma string codificada em UTF-8).

Se não forem nulos, os resultados coletados serão retornados em uma mensagem de resposta serializada no formato especificado por esse parâmetro. Consulte API de perfil do Gremlin no Neptune para obter mais informações.

Resposta

- output: um ReportAsText, do tipo: `blob` (um bloco de dados binários não interpretados).

  Um blob de texto que contém o resultado de perfil do Gremlin. Consulte API de perfil do Gremlin no Neptune para obter detalhes.

Erros

- QueryTooLargeException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException

- ConcurrentModificationException

ListGremlinQueries (ação)

O nome da CLI da AWS para essa API é: `list-gremlin-queries`.

Lista as consultas ativas do Gremlin. Consulte API de status de consulta do Gremlin para obter detalhes sobre a saída.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:GetQueryStatus nesse cluster.

Observe que a chave de condição do IAM neptune-db:QueryLanguage:Gremlin pode ser usada no documento de política para restringir o uso de consultas do Gremlin (consulte Chaves de condição disponíveis em declarações de política de acesso a dados do IAM do Neptune).

Solicitação

- includeWaiting (na CLI: `--include-waiting`): um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se definido como `TRUE`, a lista exibida incluirá consultas em espera. O padrão é `FALSE`;

Resposta

- acceptedQueryCount: um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número de consultas que foram aceitas, mas ainda não concluídas, incluindo as consultas na fila.
- queries – Uma matriz de objetos GremlinQueryStatus.

  Uma lista das consultas atuais.
- runningQueryCount: um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número de consultas do Gremlin em execução no momento.

Erros

- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ReadOnlyViolationException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

GetGremlinQueryStatus (ação)

O nome da CLI da AWS para essa API é: `get-gremlin-query-status`.

Obtém o status de uma consulta do Gremlin especificada.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:GetQueryStatus nesse cluster.

Observe que a chave de condição do IAM neptune-db:QueryLanguage:Gremlin pode ser usada no documento de política para restringir o uso de consultas do Gremlin (consulte Chaves de condição disponíveis em declarações de política de acesso a dados do IAM do Neptune).

Solicitação

- queryId (na CLI: `--query-id`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador exclusivo que identifica a consulta do Gremlin.

Resposta

- queryEvalStats – Um objeto QueryEvalStats.

  O status da avaliação da consulta do Gremlin.
- queryId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da consulta para a qual o status está sendo exibido.
- queryString: uma string, do tipo: `string` (uma string codificada em UTF-8).

  A string de consulta do Gremlin.

Erros

- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ReadOnlyViolationException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException

- ConcurrentModificationException

CancelGremlinQuery (ação)

O nome da CLI da AWS para essa API é: `cancel-gremlin-query`.

Cancela uma consulta do Gremlin. Consulte Cancelamento de consultas do Gremlin para obter mais informações.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:CancelQuery nesse cluster.

Solicitação

- queryId (na CLI: `--query-id`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador exclusivo que identifica a consulta a ser cancelada.

Resposta

- status: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O status do cancelamento

Erros

- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException

- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

Ações de consulta do openCypher:

ExecuteOpenCypherQuery (ação)

O nome da CLI da AWS para essa API é: `execute-open-cypher-query`.

Executa uma consulta do openCypher. Consulte Acessar o grafo do Netuno com o openCypher para obter mais informações.

O Neptune é compatível com a criação de aplicações de grafos usando o openCypher, atualmente uma das linguagens de consulta mais populares entre os desenvolvedores que trabalham com bancos de dados de grafos. Desenvolvedores, analistas de negócios e cientistas de dados gostam da sintaxe inspirada em SQL declarativa do openCypher porque ela oferece uma estrutura conhecida para consultar grafos de propriedades.

A linguagem openCypher foi originalmente desenvolvida pela Neo4j e, depois, passou a ser de código aberto em 2015 e contribuiu para o projeto openCypher sob uma licença de código aberto Apache 2.

Observe que, ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize uma das seguintes ações do IAM nesse cluster, dependendo da consulta:

- neptune-db:ReadDataViaQuery
- neptune-db:WriteDataViaQuery
- neptune-db:DeleteDataViaQuery

Observe também que a chave de condição do IAM neptune-db:QueryLanguage:OpenCypher pode ser usada no documento de política para restringir o uso de consultas do openCypher (consulte Chaves de condição disponíveis em declarações de política de acesso a dados do IAM do Neptune).

Solicitação

- openCypherQuery (na CLI: `--open-cypher-query`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  A string de consulta do openCypher a ser executada.
- parameters (na CLI: `--parameters`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Os parâmetros de consulta do openCypher para execução da consulta. Consulte Exemplos de consulta parametrizada do openCypher para obter mais informações.

Resposta

- results: obrigatório: um documento, do tipo `document` (um conteúdo aberto independente de protocolo representado por um modelo de dados semelhante a JSON).

  Os resultados de openCypherquery.

Erros

- QueryTooLargeException
- InvalidNumericDataException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException

- MalformedQueryException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

ExecuteOpenCypherExplainQuery (ação)

O nome da CLI da AWS para essa API é: `execute-open-cypher-explain-query`.

Executa uma solicitação `explain` do openCypher. Consulte O atributo explain do openCypher para obter mais informações.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:ReadDataViaQuery nesse cluster.

Observe que a chave de condição do IAM neptune-db:QueryLanguage:OpenCypher pode ser usada no documento de política para restringir o uso de consultas do openCypher (consulte Chaves de condição disponíveis em declarações de política de acesso a dados do IAM do Neptune).

Solicitação

- explainMode (na CLI :`--explain-mode`): obrigatório: um OpenCypherExplainMode, do tipo: `string` (uma string codificada em UTF-8).

  O modo `explain` do openCypher. Pode ser `static`, `dynamic` ou `details`.
- openCypherQuery (na CLI: `--open-cypher-query`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  A string de consulta do openCypher.
- parameters (na CLI: `--parameters`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Os parâmetros de consulta do openCypher.

Resposta

- results: obrigatório: um Blob, do tipo: `blob` (um bloco de dados binários não interpretados).

  Um blob de texto que contém os resultados `explain` do openCypher.

Erros

- QueryTooLargeException
- InvalidNumericDataException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

ListOpenCypherQueries (ação)

O nome da CLI da AWS para essa API é: `list-open-cypher-queries`.

Lista as consultas ativas do openCypher. Consulte Endpoint de status do openCypher no Neptune para obter mais informações.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:GetQueryStatus nesse cluster.

Observe que a chave de condição do IAM neptune-db:QueryLanguage:OpenCypher pode ser usada no documento de política para restringir o uso de consultas do openCypher (consulte Chaves de condição disponíveis em declarações de política de acesso a dados do IAM do Neptune).

Solicitação

- includeWaiting (na CLI: `--include-waiting`): um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Quando definido como `TRUE` e outros parâmetros não estão presentes, faz com que as informações de status das consultas em espera sejam geradas, bem como das consultas em execução.

Resposta

- acceptedQueryCount: um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número de consultas que foram aceitas, mas ainda não concluídas, incluindo as consultas na fila.
- queries – Uma matriz de objetos GremlinQueryStatus.

  Uma lista das consultas atuais do openCypher.
- runningQueryCount: um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número de consultas do openCypher em execução no momento.

Erros

- InvalidNumericDataException
- BadRequestException
- InvalidParameterException
- ClientTimeoutException

- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ReadOnlyViolationException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

GetOpenCypherQueryStatus (ação)

O nome da CLI da AWS para essa API é: `get-open-cypher-query-status`.

Recupera o status de uma consulta do openCypher especificada.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:GetQueryStatus nesse cluster.

Observe que a chave de condição do IAM neptune-db:QueryLanguage:OpenCypher pode ser usada no documento de política para restringir o uso de consultas do openCypher (consulte Chaves de condição disponíveis em declarações de política de acesso a dados do IAM do Neptune).

Solicitação

- queryId (na CLI: `--query-id`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID exclusivo da consulta do openCypher para a qual recuperar o status da consulta.

Resposta

- queryEvalStats – Um objeto QueryEvalStats.

  O status de avaliação da consulta do openCypher.

- queryId: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID exclusivo da consulta para a qual o status está sendo exibido.

- queryString: uma string, do tipo: `string` (uma string codificada em UTF-8).

  A string de consulta do openCypher.

Erros

- InvalidNumericDataException
- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ReadOnlyViolationException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

CancelOpenCypherQuery (ação)

O nome da CLI da AWS para essa API é: `cancel-open-cypher-query`.

Cancela uma consulta do openCypher especificada. Consulte Endpoint de status do openCypher no Neptune para obter mais informações.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:CancelQuery nesse cluster.

Solicitação

- queryId (na CLI: `--query-id`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID exclusivo da consulta do openCypher a ser cancelada.
- silent (na CLI: `--silent`): um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se definido como `TRUE`, faz com que o cancelamento da consulta do openCypher ocorra silenciosamente.

Resposta

- payload: um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  A carga útil de cancelamento da consulta do openCypher.
- status: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O status de cancelamento da consulta do openCypher.

Erros

- InvalidNumericDataException
- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException

- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

Estruturas de consulta:

QueryEvalStats (estrutura)

Estrutura para capturar estatísticas de consultas, como quantas consultas estão sendo executadas, aceitas ou aguardando e seus detalhes.

Campos

- cancelled: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Defina como TRUE se a consulta foi cancelada ou FALSE, caso contrário.
- elapsed: é um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número de milissegundos em que a consulta esteve em execução até agora.
- subqueries: é um documento, do tipo `document` (um conteúdo aberto independente de protocolo representado por um modelo de dados semelhante a JSON).

  O número de subconsultas nesta consulta.
- waited: é um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Indica quanto tempo a consulta esperou, em milissegundos.

GremlinQueryStatus (estrutura)

Captura o status de uma consulta do Gremlin (consulte a página API de status de consulta do Gremlin).

Campos

- queryEvalStats: é um objeto QueryEvalStats.

  As estatísticas da consulta do Gremlin.
- queryId: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da consulta do Gremlin.
- queryString: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  A string da consulta do Gremlin.

GremlinQueryStatusAttributes (estrutura)

Contém componentes de status de uma consulta do Gremlin.

Campos

- attributes: é um documento, do tipo `document` (um conteúdo aberto independente de protocolo representado por um modelo de dados semelhante a JSON).

  Atributos do status da consulta do Gremlin.
- code: é um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O código de resposta HTTP gerado pela solicitação de consulta do Gremlin.
- message: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  A mensagem de status.

APIs de carregamento em massa do plano de dados do Neptune

Ações de carregamento em massa:

- StartLoaderJob (ação)
- GetLoaderJobStatus (ação)
- ListLoaderJobs (ação)
- CancelLoaderJob (ação)

Estrutura de carregamento em massa:

- LoaderIdResult (estrutura)

StartLoaderJob (ação)

O nome da CLI da AWS para essa API é: `start-loader-job`.

Inicia um trabalho de carregador em massa do Neptune para carregar dados de um bucket do Amazon S3 em uma instância de banco de dados do Neptune. Consulte Usar o carregador em massa do Amazon Neptune para ingerir dados.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:StartLoaderJob nesse cluster.

Solicitação

- dependencies (na CLI: `--dependencies`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Trata-se de um parâmetro opcional que pode tornar uma solicitação de carga em fila dependente da conclusão bem-sucedida de um ou mais trabalhos anteriores na fila.

  O Neptune poderá colocar na fila até 64 solicitações de carga por vez se os parâmetros `queueRequest` estiverem definidos como `"TRUE"`. O parâmetro `dependencies` permite executar a solicitação enfileirada dependente da conclusão com êxito de uma ou mais solicitações anteriores especificadas na fila.

  Por exemplo, se as cargas `Job-A` e `Job-B` forem interdependentes, mas a carga `Job-C` precisar que a `Job-A` e a `Job-B` sejam concluídas antes de começar, prossiga da seguinte forma:

  1. Envie `load-job-A` e `load-job-B`, uma após a outra, em qualquer ordem e salve os load-ids.
  2. Envie `load-job-C` com os load-ids dos dois trabalhos no campo `dependencies`:

  Example

```
"dependencies" : ["(job_A_load_id)", "(job_B_load_id)"]
```

Devido ao parâmetro `dependencies`, o carregador em massa não iniciará a `Job-C` até que a `Job-A` e a `Job-B` tenham sido concluídas com êxito. Se uma delas falhar, a Job-C não será executada e o status será definido como `LOAD_FAILED_BECAUSE_DEPENDENCY_NOT_SATISFIED`.

É possível configurar vários níveis de dependência desta forma, para que a falha de um trabalho faça com que todas as solicitações direta ou indiretamente dependentes sejam canceladas.

- failOnError (na CLI: `--fail-on-error`): um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  `failOnError`: sinalizador para alternar uma interrupção completa em um erro.

  Valores permitidos: `"TRUE"`, `"FALSE"`

  Valor padrão: `"TRUE"`.

  Quando este parâmetro estiver configurado como `"FALSE"`, o carregador tentará carregar todos os dados no local especificado, ignorando as entradas com erros.

  Quando este parâmetro for definido como `"TRUE"`, o carregador será interrompido assim que encontrar um erro. Os dados carregados até esse momento serão mantidos.

- format (na CLI: `--format`): obrigatório: um formato, do tipo: `string` (uma string codificada em UTF-8).

  O formato dos dados. Para obter mais informações sobre os formatos de dados para o comando `Loader` do Neptune, consulte Formatos de dados de carga.

  Valores permitidos

  - **`csv`** para o formato de dados CSV do Gremlin.
  - **`opencypher`** para o formato de dados CSV do openCypher.
  - **`ntriples`** para o formato de dados N-Triples do RDF.
  - **`nquads`** para o formato de dados N-Quads do RDF.
  - **`rdfxml`** para o formato de dados RDF\XML do RDF.
  - **`turtle`** para o formato de dados Turtle do RDF.

- iamRoleArn (na CLI: `--iam-role-arn`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

O nome do recurso da Amazon (ARN) de um perfil do IAM a ser assumido pela instância de banco de dados do Neptune para acesso ao bucket do S3. O ARN do perfil do IAM fornecido aqui deve ser anexado ao cluster de banco de dados (consulte Adicionar o perfil do IAM a um cluster do Amazon Neptune).

- mode (na CLI: `--mode`): um modo, do tipo: `string` (uma string codificada em UTF-8).

  Modo do trabalho de carga.

  Valores permitidos: `RESUME`, `NEW`, `AUTO`.

  Valor padrão: `AUTO`.

  - `RESUME`: no modo RESUME, o carregador procura uma carga anterior dessa origem e, se encontrar, retomará esse trabalho de carga. Se nenhum trabalho de carga anterior for encontrado, o carregador será interrompido.

    O carregador evita recarregar arquivos que foram carregados com êxito em um trabalho anterior. Ele só tenta processar arquivos com falha. Se você descartou anteriormente dados carregados do cluster do Neptune, esses dados não serão recarregados nesse modo. Se um trabalho de carga anterior carregou todos os arquivos da mesma origem com êxito, nada será recarregado, e o recarregador exibirá êxito.
  - `NEW`: no modo NEW, cria uma solicitação de carga, independentemente de quaisquer cargas anteriores. É possível usar esse modo para recarregar todos dados de uma origem depois de eliminar dados carregados anteriormente do cluster do Neptune ou de carregar novos dados disponíveis na mesma origem.
  - `AUTO`: no modo AUTO, o carregador procura um trabalho de carga anterior da mesma origem e, se encontrar, retomará esse trabalho, assim como no modo `RESUME`.

    Se o carregador não encontrar um trabalho de carga anterior da mesma origem, ele carregará todos os dados da origem, assim como no modo `NEW`.
- parallelism (na CLI: `--parallelism`): um paralelismo, do tipo: `string` (uma string codificada em UTF-8).

  O parâmetro `parallelism` opcional que pode ser definido para reduzir o número de threads usados pelo processo de carregamento em massa.

  Valores permitidos:

- `LOW`: o número de threads usados é o número de vCPUs dividido por oito.
- `MEDIUM`: o número de threads usados é o número de vCPUs disponíveis dividido por dois.
- `HIGH`: o número de threads usados é equivalente ao número de vCPUs disponíveis.
- `OVERSUBSCRIBE`: o número de threads usados é o número de vCPUs disponíveis multiplicado por dois. Se esse valor for usado, o carregador em massa usará todos os recursos disponíveis.

  No entanto, isso não significa que a configuração `OVERSUBSCRIBE` resulte em 100% de utilização da CPU. Como a operação de carga está limitada à E/S, a maior utilização esperada da CPU está na faixa de 60% a 70%.

Valor padrão: `HIGH`

Às vezes, a configuração `parallelism` pode causar um deadlock entre os threads ao carregar dados do openCypher. Quando isso acontece, o Neptune gera o erro `LOAD_DATA_DEADLOCK`. Geralmente, é possível corrigir o problema definindo uma configuração `parallelism` mais baixa e tentando novamente o comando de carregamento.

- parserConfiguration (na CLI: `--parser-configuration`): uma matriz de mapa dos pares de valor-chave em que:

  Cada chave é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Cada valor é uma string, do tipo: `string` (uma string codificada em UTF-8).

  `parserConfiguration`: um objeto opcional com valores de configuração de analisador adicional. Os parâmetros filho também são opcionais:

  - **`namedGraphUri`**: o grafo padrão para todos os formatos do RDF quando nenhum grafo for especificado (para formatos sem quadrantes e entradas NQUAD sem grafo).

    O padrão é `https://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`.
  - **`baseUri`**: o URI base para formatos RDF/XML e Turtle.

    O padrão é `https://aws.amazon.com/neptune/default`.
  - **`allowEmptyStrings`**: é necessário que os usuários do Gremlin possam transmitir valores de string vazios (“”) como propriedades de nós e bordas ao carregar dados CSV. Se `allowEmptyStrings` estiver definido como `false` (o padrão), essas strings vazias serão tratadas como nulas e não serão carregadas.

Se `allowEmptyStrings` estiver definido como `true`, o carregador tratará strings vazias como valores de propriedade válidos e as carregará adequadamente.

- queueRequest (na CLI: `--queue-request`): um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Trata-se de um parâmetro de sinalizador opcional que indica se a solicitação de carga pode ser colocada em fila ou não.

  Não é necessário esperar um trabalho de carga ser concluído antes de emitir o próximo, porque o Neptune pode colocar na fila até 64 trabalhos por vez, desde que os parâmetros `queueRequest` estejam todos definidos como `"TRUE"`. A ordem da fila dos trabalhos será do tipo FIFO (first-in-first-out, primeiro a entrar, primeiro a sair).

  Se o parâmetro `queueRequest` for omitido ou definido como `"FALSE"`, a solicitação de carga falhará se outro trabalho de carga já estiver em execução.

  Valores permitidos: `"TRUE"`, `"FALSE"`

  Valor padrão: `"FALSE"`.

- s3BucketRegion (na CLI: `--s-3-bucket-region`): obrigatório: uma S3BucketRegion, do tipo: `string` (uma string codificada em UTF-8).

  A região da Amazon do bucket do S3. Deve corresponder à região da Amazon do cluster de banco de dados.

- source (na CLI: `--source`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O parâmetro `source` aceita um URI do S3 que identifica um único arquivo, vários arquivos, uma pasta ou várias pastas. O Neptune carrega todos os arquivos de dados em qualquer pasta especificada.

  O URI pode estar em qualquer um dos seguintes formatos.

  - `s3://(bucket_name)/(object-key-name)`
  - `https://s3.amazonaws.com/(bucket_name)/(object-key-name)`
  - `https://s3.us-east-1.amazonaws.com/(bucket_name)/(object-key-name)`

O elemento `object-key-name` do URI é equivalente ao parâmetro prefix em uma chamada da API ListObjects do S3. Ele identifica todos os objetos no bucket do S3 especificado cujos nomes começam com esse prefixo. Pode ser um único arquivo ou pasta, ou vários arquivos e/ou pastas.

A pasta ou as pastas especificadas podem conter vários arquivos de vértice e vários arquivos de borda.

- updateSingleCardinalityProperties (na CLI: `--update-single-cardinality-properties`): um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  `updateSingleCardinalityProperties` é um parâmetro opcional que controla como o carregador em massa trata um novo valor das propriedades de vértice de cardinalidade única ou de borda. Isso não é compatível com o carregamento de dados do openCypher.

  Valores permitidos: `"TRUE"`, `"FALSE"`

  Valor padrão: `"FALSE"`.

  Por padrão, ou quando `updateSingleCardinalityProperties` está explicitamente definido como `"FALSE"`, o carregador trata um novo valor como um erro, porque ele viola a cardinalidade única.

  Quando `updateSingleCardinalityProperties` está definido como `"TRUE"`, por outro lado, o carregador em massa substitui o valor existente pelo novo. Se vários valores de propriedade de vértice de cardinalidade única ou de ponto forem fornecidos nos arquivos de origem que estão sendo carregados, o valor final quando o carregamento em massa terminar poderá ser qualquer um desses novos valores. O carregador só garante que o valor existente tenha sido substituído por um dos novos.

- userProvidedEdgeIds (na CLI: `--user-provided-edge-ids`): um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Esse parâmetro é necessário somente ao carregar dados do openCypher que contêm IDs de relacionamento. Ele deve ser incluído e definido como `True` quando os IDs de relacionamento do openCypher forem fornecidos explicitamente nos dados de carregamento (recomendado).

  Quando `userProvidedEdgeIds` está ausente ou definido como `True`, uma coluna `:ID` deve estar presente em cada arquivo de relacionamento no carregamento.

Quando `userProvidedEdgeIds` está presente e definido como `False`, os arquivos de relacionamento no carregamento não devem conter uma coluna `:ID`. Em vez disso, o carregador do Neptune gera automaticamente um ID para cada relacionamento.

É útil fornecer IDs de relacionamento explicitamente para que o carregador possa retomar o carregamento após a correção do erro nos dados CSV, sem precisar recarregar nenhum relacionamento que já tenha sido carregado. Se os IDs de relacionamento não tiverem sido atribuídos explicitamente, o carregador não poderá retomar um carregamento com falha se for necessário corrigir algum arquivo de relacionamento. Nesse caso, ele deverá carregar todos os relacionamentos.

Resposta

- payload: obrigatório: é uma matriz de mapa dos pares de valor-chave em que:

  Cada chave é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Cada valor é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Contém um par de nome-valor `loadId` que fornece um identificador para a operação de carregamento.
- status: obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O código HTTP que indica o status do trabalho de carregamento.

Erros

- BadRequestException
- InvalidParameterException
- BulkLoadIdNotFoundException
- ClientTimeoutException
- LoadUrlAccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- InternalFailureException

- S3Exception
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

GetLoaderJobStatus (ação)

O nome da CLI da AWS para essa API é: `get-loader-job-status`.

Obtém informações de status sobre um trabalho de carregamento especificado. O Neptune mantém o controle dos 1.024 trabalhos de carregamento em massa mais recentes e armazena somente os últimos 10 mil detalhes de erro por trabalho.

Consulte API Get-Status do carregador do Neptune para obter mais informações.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:GetLoaderJobStatus nesse cluster.

Solicitação

- details (na CLI: `--details`): um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Sinalizador que indica se é necessário ou não incluir detalhes além do status geral (`TRUE` ou `FALSE`; o padrão é `FALSE`).
- errors (na CLI: `--errors`): um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Sinalizador que indica se é necessário ou não incluir uma lista de erros encontrados (`TRUE` ou `FALSE`; o padrão é `FALSE`).

  A lista de erros é paginada. Os parâmetros `page` e `errorsPerPage` permitem que você pagine por todos os erros.
- errorsPerPage (na CLI :`--errors-per-page`): um PositiveInteger, do tipo: `integer` (um valor inteiro assinado de 32 bits), pelo menos 1 ?st?.

O número de erros exibidos em cada página (um número inteiro positivo; o padrão é `10`). Válido apenas com o parâmetro `errors` definido como `TRUE`.

- loadId (na CLI: `--load-id`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID do trabalho de carregamento do qual obter o status.

- page (na CLI :`--page`): um PositiveInteger, do tipo: `integer` (um valor inteiro assinado de 32 bits), pelo menos 1 ?st?.

  O número de páginas de erros (um número inteiro positivo; o padrão é `1`). Válido apenas quando o parâmetro `errors` é definido como `TRUE`.

Resposta

- payload: obrigatório: um documento, do tipo `document` (um conteúdo aberto independente de protocolo representado por um modelo de dados semelhante a JSON).

  Informações de status sobre o trabalho de carregamento, em um layout que pode ser semelhante ao seguinte:

  Example

```
{
          "status" : "200 OK",
          "payload" : {
            "feedCount" : [
              {
                "LOAD_FAILED" : (number)
              }
            ],
            "overallStatus" : {
              "fullUri" : "s3://(bucket)/(key)",
              "runNumber" : (number),
              "retryNumber" : (number),
              "status" : "(string)",
              "totalTimeSpent" : (number),
              "startTime" : (number),
              "totalRecords" : (number),
              "totalDuplicates" : (number),
              "parsingErrors" : (number),
```

```
            "datatypeMismatchErrors" : (number),
            "insertErrors" : (number),
          },
          "failedFeeds" : [
            {
              "fullUri" : "s3://(bucket)/(key)",
              "runNumber" : (number),
              "retryNumber" : (number),
              "status" : "(string)",
              "totalTimeSpent" : (number),
              "startTime" : (number),
              "totalRecords" : (number),
              "totalDuplicates" : (number),
              "parsingErrors" : (number),
              "datatypeMismatchErrors" : (number),
              "insertErrors" : (number),
            }
          ],
          "errors" : {
            "startIndex" : (number),
            "endIndex" : (number),
            "loadId" : "(string),
            "errorLogs" : [ ]
          }
        }
      }
```

- status: obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O código de resposta HTTP da solicitação.

Erros

- BadRequestException
- InvalidParameterException
- BulkLoadIdNotFoundException
- ClientTimeoutException
- LoadUrlAccessDeniedException
- IllegalArgumentException
- TooManyRequestsException

- UnsupportedOperationException
- InternalFailureException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

ListLoaderJobs (ação)

O nome da CLI da AWS para essa API é: `list-loader-jobs`.

Recupera uma lista de `loadIds` de todos os trabalhos ativos do carregador.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:ListLoaderJobs nesse cluster.

Solicitação

- includeQueuedLoads (na CLI: `--include-queued-loads`): um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Um parâmetro opcional que pode ser usado para excluir os IDs de solicitações de carregamento na fila quando uma lista de IDs de carregamento for solicitada definindo o parâmetro como `FALSE`. O valor padrão é `TRUE`.
- limit (na CLI :`--limit`): um ListLoaderJobsInputLimitInteger, do tipo: `integer` (um valor inteiro assinado de 32 bits), não menos que 1 ou mais que 100 ?st?s.

  O número de IDs de carregamento a serem listados. Deve ser um número inteiro positivo maior que zero e menor que `100` (o padrão).

Resposta

- payload – Obrigatório: um objeto LoaderIdResult.

  A lista de IDs de trabalho solicitada.
- status: obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

Exibe o status da solicitação da lista de trabalhos.

Erros

- UnsupportedOperationException
- BadRequestException
- InvalidParameterException
- BulkLoadIdNotFoundException
- InternalFailureException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- LoadUrlAccessDeniedException
- IllegalArgumentException
- TooManyRequestsException

CancelLoaderJob (ação)

O nome da CLI da AWS para essa API é: `cancel-loader-job`.

Cancela um trabalho de carregamento especificado. Essa é uma solicitação `DELETE` HTTP. Consulte API Get-Status do carregador do Neptune para obter mais informações.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:CancelLoaderJob nesse cluster.

Solicitação

- loadId (na CLI: `--load-id`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID do trabalho de carregamento a ser excluído.

Resposta

- status: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O status do cancelamento.

Erros

- BadRequestException
- InvalidParameterException
- BulkLoadIdNotFoundException
- ClientTimeoutException
- LoadUrlAccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- InternalFailureException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

Estrutura de carregamento em massa:

LoaderIdResult (estrutura)

Contém uma lista de IDs de carregamento.

Campos

- loadIds: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de IDs de carregamento.

API de plano de dados dos fluxos do Neptune

Ações de acesso aos fluxos:

- GetPropertygraphStream (ação)

Estruturas de dados de fluxos:

- PropertygraphRecord (estrutura)
- PropertygraphData (estrutura)

GetPropertygraphStream (ação)

O nome da CLI da AWS para essa API é: `get-propertygraph-stream`.

Obtém um fluxo para um grafo de propriedades.

Com o atributo de fluxos `GetPropertygraphStream`, é possível gerar uma sequência completa de entradas no log de alterações que registram todas as alterações feitas nos dados de grafos à medida que elas ocorrem. O permite coletar essas entradas do log de alterações para um grafo de propriedades.

O atributo de fluxos do Neptune precisa estar habilitado no DBcluster do Neptune. Para habilitar fluxos, defina o parâmetro do cluster de banco neptune_streams como 1.

Consulte Capturar alterações do grafo em tempo real usando os fluxos do Neptune.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:GetStreamRecords nesse cluster.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize uma das seguintes ações do IAM, dependendo da consulta:

Observe que é possível restringir as consultas de grafos de propriedades usando as seguintes chaves de contexto do IAM:

- neptune-db:QueryLanguage:Gremlin
- neptune-db:QueryLanguage:OpenCypher

Consulte Chaves de condição disponíveis em declarações de política de acesso a dados do IAM do Neptune.

Solicitação

- commitNum (na CLI: `--commit-num`): um valor longo, do tipo: `long` (um valor inteiro assinado de 64 bits).

  O número de confirmação do registro inicial a ser lido no fluxo do log de alterações. Esse parâmetro é necessário quando `iteratorType` é `AT_SEQUENCE_NUMBER` ou `AFTER_SEQUENCE_NUMBER` e ignorado quando `iteratorType` é `TRIM_HORIZON` ou`LATEST`.
- encoding (na CLI: `--encoding`): uma codificação, do tipo: `string` (uma string codificada em UTF-8).

  Se definido como TRUE, o Neptune comprime a resposta usando a codificação gzip.
- iteratorType (na CLI: `--iterator-type`): um IteratorType, do tipo: `string` (uma string codificada em UTF-8).

  Pode ser um dos seguintes:

  - `AT_SEQUENCE_NUMBER`: indica que a leitura deve começar a partir do número de sequência de eventos especificado em conjunto pelos parâmetros `commitNum` e opNum.
  - `AFTER_SEQUENCE_NUMBER`: indica que a leitura deve começar logo após o número de sequência de eventos especificado em conjunto pelos parâmetros `commitNum` e opNum.
  - `TRIM_HORIZON`: indica que a leitura deve começar no último registro não truncado no sistema, que é o registro mais antigo não expirado (ainda não excluído) no fluxo de logs de alterações.
  - `LATEST`: indica que a leitura deve começar no registro mais recente no sistema, que é o registro mais recente não expirado (ainda não excluído) no fluxo de logs de alterações.
- limit (na CLI: `--limit`): um GetPropertygraphStreamInputLimitLong, do tipo: `long` (um valor inteiro assinado de 64 bits), não menos que 1 ou mais que 100.000 ?st?s.

  Especifica o número máximo de registros a serem retornados. Há também um limite de tamanho de 10 MB na resposta que não pode ser modificado e que tem precedência sobre o número de registros especificado no parâmetro `limit`. A resposta incluirá um registro de violação de limite se o limite de 10 MB tiver sido atingido.

  O intervalo para `limit` é de 1 a 100.000, com um padrão de 10.
- opNum (na CLI: `--op-num`): um valor longo, do tipo: `long` (um valor inteiro assinado de 64 bits).

O número de sequência da operação dentro da confirmação especificada da qual começar a ler nos dados do fluxo do log de alterações. O padrão é 1.

Resposta

- format: obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  o formato da serialização dos registros de alterações que estão sendo retornados. Atualmente, o único valor compatível é `PG_JSON`.
- lastEventId: obrigatório: é uma matriz de mapa dos pares de chave-valor em que:

  Cada chave é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Cada valor é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Identificador da sequência da última alteração na resposta do fluxo.

  Um ID de evento é composto de dois campos: um `commitNum` que identifica uma transação que alterou o grafo, e um `opNum` que identifica uma operação específica dentro dessa transação:

  Example

  ```
  "eventId": {
            "commitNum": 12,
            "opNum": 1
          }
  ```
- lastTrxTimestampInMillis: obrigatório: um valor longo, do tipo: `long` (um valor inteiro assinado de 64 bits).

  A hora em que a confirmação da transação foi solicitada, em milissegundos a partir do epoch do Unix.
- records: obrigatório: uma matriz de objetos PropertygraphRecord.

  Uma matriz de registros serializados do fluxo de log de alterações incluídos na resposta.
- totalRecords: obrigatório: um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número total de registros na resposta.

Erros

- UnsupportedOperationException
- ExpiredStreamException
- InvalidParameterException
- MemoryLimitExceededException
- StreamRecordsNotFoundException
- ClientTimeoutException
- PreconditionsFailedException
- ThrottlingException
- ConstraintViolationException
- InvalidArgumentException
- IllegalArgumentException
- TooManyRequestsException

Estruturas de dados de fluxos:

PropertygraphRecord (estrutura)

Estrutura de um registro de grafo de propriedades.

Campos

- commitTimestampInMillis: é obrigatório: um valor longo, do tipo: `long` (um valor inteiro assinado de 64 bits).

  A hora em que a confirmação da transação foi solicitada, em milissegundos a partir do epoch do Unix.

- data: é obrigatório: um objeto PropertygraphData.

  O registro de alteração serializado do Gremlin ou do openCypher.

- eventId: é obrigatório: é uma matriz de mapa dos pares de chave-valor em que:

  Cada chave é uma string, do tipo: `string` (uma string codificada em UTF-8).

Cada valor é uma string, do tipo: `string` (uma string codificada em UTF-8).

O identificador da sequência do registro de alteração do fluxo.

- isLastOp: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Presente somente se essa operação for a última da transação. Se estiver presente, será definido como verdadeiro. É útil para garantir que uma transação inteira seja consumida.
- op: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  A operação que criou a alteração.

PropertygraphData (estrutura)

Um registro de alteração do Gremlin ou do openCypher.

Campos

- from: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Se esta for uma borda (tipo = `e`), o ID do vértice `from` correspondente ou do nó de origem.
- id: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID do elemento do Gremlin ou do openCypher.
- key: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome da propriedade. Para rótulos de elementos, é `label`.
- to: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Se esta for uma borda (tipo = `e`), o ID do vértice `to` correspondente ou do nó de destino.
- type: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo desse elemento do Gremlin ou do openCypher. Deve ser um dos seguintes:
  - **v1**: rótulo de vértice para Gremlin ou o rótulo de nó para openCypher.
  - **vp**: propriedades de vértice para Gremlin ou propriedades de nós para openCypher.
  - **e**: borda e rótulo de borda para Gremlin ou relacionamento e tipo de relacionamento para openCypher.
  - **ep**: propriedades de borda para Gremlin ou propriedades de relacionamento para openCypher.

- value: é obrigatório: um documento, do tipo `document` (um conteúdo aberto independente de protocolo representado por um modelo de dados semelhante a JSON).

  Trata-se de um objeto JSON que contém um campo de valor para o próprio valor, e um campo de tipo de dados para o tipo de dados JSON desse valor:

  Example

```
"value": {
          "value": "(the new value"),
          "dataType": "(the JSON datatypenew value")
        }
```

Estatísticas do plano de dados do Neptune e APIs de resumo de grafos

Ações estatísticas do grafo de propriedades:

- GetPropertygraphStatistics (ação)
- ManagePropertygraphStatistics (ação)
- DeletePropertygraphStatistics (ação)
- GetPropertygraphSummary (ação)

Estruturas estatísticas:

- Estatísticas (estrutura)
- StatisticsSummary (estrutura)
- DeleteStatisticsValueMap (estrutura)
- RefreshStatisticsIdMap (estrutura)
- NodeStructure (estrutura)
- EdgeStructure (estrutura)
- SubjectStructure (estrutura)
- PropertygraphSummaryValueMap (estrutura)
- PropertygraphSummary (estrutura)

GetPropertygraphStatistics (ação)

O nome da CLI da AWS para essa API é: `get-propertygraph-statistics`.

Obtém estatísticas do grafo de propriedades (Gremlin e openCypher).

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:GetStatisticsStatus nesse cluster.

Solicitação

- Sem parâmetros de solicitação.

Resposta

- payload – Obrigatório: um objeto Estatísticas.

  Estatísticas para dados de grafos de propriedades.
- status: obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O código de retorno HTTP da solicitação. Se a solicitação for bem-sucedida, o código será 200. Consulte Códigos de erro comuns para solicitação de estatísticas do DFE para obter uma lista de erros comuns.

Erros

- BadRequestException
- InvalidParameterException
- StatisticsNotAvailableException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- PreconditionsFailedException

- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

ManagePropertygraphStatistics (ação)

O nome da CLI da AWS para essa API é: `manage-propertygraph-statistics`.

Gerencia a geração e o uso de estatísticas de grafos de propriedades.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:ManageStatistics nesse cluster.

Solicitação

- mode (na CLI: `--mode`): um StatisticsAutoGenerationMode, do tipo: `string` (uma string codificada em UTF-8).

  O modo de geração de estatísticas. Um destes: `DISABLE_AUTOCOMPUTE`, `ENABLE_AUTOCOMPUTE` ou `REFRESH`, o último dos quais aciona manualmente a geração de estatísticas do DFE.

Resposta

- payload – Um objeto RefreshStatisticsIdMap.

  É exibido apenas para o modo de atualização.
- status: obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O código de retorno HTTP da solicitação. Se a solicitação for bem-sucedida, o código será 200.

Erros

- BadRequestException
- InvalidParameterException

- StatisticsNotAvailableException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- PreconditionsFailedException
- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

DeletePropertygraphStatistics (ação)

O nome da CLI da AWS para essa API é: `delete-propertygraph-statistics`.

Exclui estatísticas dos dados (grafo de propriedades) do Gremlin e do openCypher.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:DeleteStatistics nesse cluster.

Solicitação

- Sem parâmetros de solicitação.

Resposta

- payload – Um objeto DeleteStatisticsValueMap.

  A carga útil da exclusão.
- status: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O status de cancelamento.
- statusCode: um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

O código de resposta HTTP: 200 se a exclusão for bem-sucedida ou 204 se não houver estatísticas para excluir.

Erros

- BadRequestException
- InvalidParameterException
- StatisticsNotAvailableException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- PreconditionsFailedException
- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

GetPropertygraphSummary (ação)

O nome da CLI da AWS para essa API é: `get-propertygraph-summary`.

Obtém um resumo de um grafo de propriedades.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:GetGraphSummary nesse cluster.

Solicitação

- mode (na CLI: `--mode`): um GraphSummaryType, do tipo: `string` (uma string codificada em UTF-8).

  O modo pode assumir um dos dois valores: `BASIC` (o padrão) e `DETAILED`.

Resposta

- payload – Um objeto PropertygraphSummaryValueMap.

  Carga útil que contém a resposta resumida do grafo de propriedades.
- statusCode: um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O código de retorno HTTP da solicitação. Se a solicitação for bem-sucedida, o código será 200.

Erros

- BadRequestException
- InvalidParameterException
- StatisticsNotAvailableException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- PreconditionsFailedException
- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

Estruturas estatísticas:

Estatísticas (estrutura)

Contém informações estatísticas. O mecanismo DFE usa informações sobre os dados no grafo do Neptune para fazer compensações efetivas ao planejar a execução da consulta. Essas informações assumem a forma de estatísticas que incluem os chamados conjuntos de características e estatísticas de predicados que podem orientar o planejamento de consultas. Consulte Gerenciar estatísticas a serem utilizadas pelo DFE do Neptune.

Campos

- active: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica se a geração automática de estatísticas do DFE está ou não habilitada.
- autoCompute: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Indica se a geração automática de estatísticas está habilitada ou não.
- date: é um SyntheticTimestamp_date_time, do tipo: `string` (uma string codificada em UTF-8).

  A hora UTC na qual as estatísticas do DFE foram geradas mais recentemente.
- note: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma observação sobre problemas quando as estatísticas são inválidas.
- signatureInfo: é um objeto StatisticsSummary.

  Uma estrutura StatisticsSummary que contém:
  - `signatureCount`: o número total de assinaturas em todos os conjuntos de características.
  - `instanceCount`: o número total de instâncias do conjunto de características.
  - `predicateCount`: o número total de predicados exclusivos.
- statisticsId: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Relata o ID da execução atual da geração de estatísticas. Um valor de -1 indica que nenhuma estatística foi gerada.

StatisticsSummary (estrutura)

Informações sobre os conjuntos de características gerados nas estatísticas.

Campos

- instanceCount: é um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número total de instâncias do conjunto de características.
- predicateCount: é um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número total de predicados exclusivos.
- signatureCount: é um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

O número total de assinaturas em todos os conjuntos de características.

DeleteStatisticsValueMap (estrutura)

A carga útil de DeleteStatistics.

Campos

- active: é um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  O status atual das estatísticas.
- statisticsId: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da geração de estatísticas que está ocorrendo no momento.

RefreshStatisticsIdMap (estrutura)

Estatísticas do modo `REFRESH`.

Campos

- statisticsId: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da geração de estatísticas que está ocorrendo no momento.

NodeStructure (estrutura)

Uma estrutura de nó.

Campos

- count: é um valor longo, do tipo: `long` (um valor inteiro assinado de 64 bits).

  Número de nós que têm essa estrutura específica.
- distinctOutgoingEdgeLabels: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de rótulos de borda de saída distintos presentes nessa estrutura específica.
- nodeProperties: é uma string, do tipo: `string` (uma string codificada em UTF-8).

Uma lista das propriedades dos nós presentes nessa estrutura específica.

EdgeStructure (estrutura)

Uma estrutura de borda.

Campos

- count: é um valor longo, do tipo: `long` (um valor inteiro assinado de 64 bits).

  O número de bordas que têm essa estrutura específica.
- edgeProperties: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista das propriedades das bordas presentes nessa estrutura específica.

SubjectStructure (estrutura)

Uma estrutura de assunto.

Campos

- count: é um valor longo, do tipo: `long` (um valor inteiro assinado de 64 bits).

  Número de ocorrências dessa estrutura específica.
- predicates: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista dos predicados presentes nessa estrutura específica.

PropertygraphSummaryValueMap (estrutura)

Carga útil da resposta resumida do grafo de propriedades.

Campos

- graphSummary: é um objeto PropertygraphSummary.

  O resumo do grafo.
- lastStatisticsComputationTime: é um SyntheticTimestamp_date_time, do tipo: `string` (uma string codificada em UTF-8).

A data e hora, no formato ISO 8601, da hora em que o Neptune calculou as estatísticas pela última vez.

- version: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  A versão dessa resposta de resumo do grafo.

PropertygraphSummary (estrutura)

A API de resumo do grafo gera uma lista somente leitura de rótulos de nós e bordas e chaves de propriedade, junto com contagens de nós, bordas e propriedades. Consulte Resposta do resumo de um grafo de propriedades (PG).

Campos

- edgeLabels: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de rótulos de bordas distintos no grafo.
- edgeProperties: são objetos LongValuedMap. É uma matriz de mapa dos pares de chave-valor em que:

  Cada chave é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Cada valor é um valor longo, do tipo: `long` (um valor inteiro assinado de 64 bits).

  Uma lista de propriedades de bordas distintas no grafo, junto com a contagem de bordas em que cada propriedade é usada.
- edgeStructures: é uma matriz de objetos EdgeStructure.

  Esse campo só está presente quando o modo solicitado é `DETAILED`. Ele contém uma lista de estruturas de bordas.
- nodeLabels: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma lista de rótulos de nós distintos no grafo.
- nodeProperties: são objetos LongValuedMap. É uma matriz de mapa dos pares de chave-valor em que:

  Cada chave é uma string, do tipo: `string` (uma string codificada em UTF-8).

Cada valor é um valor longo, do tipo: `long` (um valor inteiro assinado de 64 bits).

O número de propriedades de nós distintas no grafo.

- nodeStructures: é uma matriz de objetos NodeStructure.

  Esse campo só está presente quando o modo solicitado é `DETAILED`. Ele contém uma lista de estruturas de nós.

- numEdgeLabels: é um valor longo, do tipo: `long` (um valor inteiro assinado de 64 bits).

  O número de rótulos de bordas distintos no grafo.

- numEdgeProperties: é um valor longo, do tipo: `long` (um valor inteiro assinado de 64 bits).

  O número de propriedades de bordas distintas no grafo.

- numEdges: é um valor longo, do tipo: `long` (um valor inteiro assinado de 64 bits).

  O número de bordas no grafo.

- numNodeLabels: é um valor longo, do tipo: `long` (um valor inteiro assinado de 64 bits).

  O número de rótulos de nós distintos no grafo.

- numNodeProperties: é um valor longo, do tipo: `long` (um valor inteiro assinado de 64 bits).

  Uma lista de propriedades de nós distintas no grafo, junto com a contagem de nós em que cada propriedade é usada.

- numNodes: é um valor longo, do tipo: `long` (um valor inteiro assinado de 64 bits).

  O número de nós no grafo.

- totalEdgePropertyValues: é um valor longo, do tipo: `long` (um valor inteiro assinado de 64 bits).

  O número total de usos de todas as propriedades de bordas.

- totalNodePropertyValues: é um valor longo, do tipo: `long` (um valor inteiro assinado de 64 bits).

  O número total de usos de todas as propriedades de nós.

API de processamento de dados do Neptune ML

Ações de processamento de dados:

- StartMLDataProcessingJob (ação)
- ListMLDataProcessingJobs (ação)
- GetMLDataProcessingJob (ação)
- CancelMLDataProcessingJob (ação)

Estruturas de uso geral de ML:

- MlResourceDefinition (estrutura)
- MlConfigDefinition (estrutura)

StartMLDataProcessingJob (ação)

O nome da CLI da AWS para essa API é: `start-ml-data-processing-job`.

Cria um trabalho de processamento de dados do Neptune ML para processar os dados de grafos exportados do Neptune para treinamento. Consulte O comando `dataprocessing`.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:StartMLModelDataProcessingJob nesse cluster.

Solicitação

- configFileName (na CLI: `--config-file-name`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um arquivo de especificação de dados que descreve como carregar os dados de grafos exportados para treinamento. O arquivo é gerado automaticamente pelo kit de ferramentas de exportação do Neptune. O padrão é `training-data-configuration.json`.
- id (na CLI: `--id`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um identificador exclusivo do novo trabalho. O padrão é um UUID gerado automaticamente.
- inputDataS3Location (na CLI: `--input-data-s3-location`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O URI do local do Amazon S3 em que você deseja que o SageMaker baixe os dados necessários para executar o trabalho de processamento de dados.

- modelType (na CLI: `--model-type`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um dos dois tipos de modelo que o Neptune ML aceita no momento: modelos de grafos heterogêneos (`heterogeneous`) e grafo de conhecimento (`kge`). O padrão é nenhum. Se não for especificado, o Neptune ML escolherá o tipo de modelo automaticamente com base nos dados.

- neptuneIamRoleArn (na CLI: `--neptune-iam-role-arn`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do recurso da Amazon (ARN) de um perfil do IAM que o SageMaker pode assumir para executar tarefas em seu nome. Deve estar listado no grupo de parâmetros do cluster de banco de dados ou ocorrerá um erro.

- previousDataProcessingJobId (na CLI: `--previous-data-processing-job-id`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID de um trabalho de processamento de dados concluído executado em uma versão anterior dos dados.

- processedDataS3Location (na CLI: `--processed-data-s3-location`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O URI do local do Amazon S3 onde você deseja que o SageMaker salve os resultados do trabalho de processamento de dados.

- processingInstanceType (na CLI: `--processing-instance-type`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo de instância de ML usada durante o processamento de dados. A memória deve ser grande o suficiente para armazenar o conjunto de dados processado. O padrão é o menor tipo ml.r5 cuja memória é dez vezes maior que o tamanho dos dados de grafos exportados no disco.

- processingInstanceVolumeSizeInGB (na CLI: `--processing-instance-volume-size-in-gb`): um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O tamanho do volume do disco da instância de processamento. Tanto os dados de entrada quanto os dados processados são armazenados em disco, portanto, o tamanho do volume deve ser grande o suficiente para conter os dois conjuntos de dados. O padrão é 0. Se não for especificado ou for 0, o Neptune ML escolherá o tamanho do volume automaticamente com base no tamanho dos dados.

- processingTimeOutInSeconds (na CLI: `--processing-time-out-in-seconds`): um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O tempo limite em segundos do trabalho de processamento de dados. O padrão é 86.400 (1 dia).

- s3OutputEncryptionKMSKey (na CLI: `--s-3-output-encryption-kms-key`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  A chave do Amazon Key Management Service (Amazon KMS) que o SageMaker usa para criptografar a saída do trabalho de processamento. O padrão é nenhum.

- sagemakerIamRoleArn (na CLI: `--sagemaker-iam-role-arn`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN de um perfil do IAM para execução do SageMaker. Deve estar listado no grupo de parâmetros do cluster de banco de dados ou ocorrerá um erro.

- securityGroupIds (na CLI: `--security-group-ids`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Os IDs dos grupos de segurança da VPC. O padrão é Nenhum.

- subnets (na CLI: `--subnets`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Os IDs das sub-redes na VPC do Neptune. O padrão é Nenhum.

- volumeEncryptionKMSKey (na CLI: `--volume-encryption-kms-key`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  A chave do Amazon Key Management Service (Amazon KMS) que o SageMaker utiliza para criptografar dados no volume de armazenamento anexado às instâncias de computação de ML que executam o trabalho de treinamento. O padrão é Nenhum.

Resposta

- arn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN do trabalho de processamento de dados.

- creationTimeInMillis: é um valor longo, do tipo: `long` (um valor inteiro assinado de 64 bits).

  O tempo necessário para criar o trabalho de processamento, em milissegundos.

- id: uma string, do tipo: `string` (uma string codificada em UTF-8).

O ID exclusivo do novo trabalho de processamento de dados.

Erros

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

ListMLDataProcessingJobs (ação)

O nome da CLI da AWS para essa API é: `list-ml-data-processing-jobs`.

Exibe uma lista de trabalhos de processamento de dados do Neptune ML. Consulte Listar trabalhos de processamento de dados ativos usando o comando dataprocessing do Neptune ML.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:ListMLDataProcessingJobs nesse cluster.

Solicitação

- maxItems (na CLI :`--max-items`): um ListMLDataProcessingJobsInputMaxItemsInteger, do tipo: `integer` (um valor inteiro assinado de 32 bits), no mínimo 1 e até 1.024 ?st?s.

  O número máximo de itens a serem exibidos (de 1 a 1.024; o padrão é 10).
- neptuneIamRoleArn (na CLI: `--neptune-iam-role-arn`): uma string, do tipo: `string` (uma string codificada em UTF-8).

O ARN de um perfil do IAM que concede ao Neptune acesso aos recursos do SageMaker e do Amazon S3. Deve estar listado no grupo de parâmetros do cluster de banco de dados ou ocorrerá um erro.

Resposta

- ids: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma página que lista os IDs dos trabalhos de processamento de dados.

Erros

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

GetMLDataProcessingJob (ação)

O nome da CLI da AWS para essa API é: `get-ml-data-processing-job`.

Recupera informações sobre um trabalho de processamento de dados especificado. Consulte O comando `dataprocessing`.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:neptune-db:GetMLDataProcessingJobStatus nesse cluster.

Solicitação

- id (na CLI: `--id`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador exclusivo do trabalho de processamento de dados a ser recuperado.
- neptuneIamRoleArn (na CLI: `--neptune-iam-role-arn`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN de um perfil do IAM que concede ao Neptune acesso aos recursos do SageMaker e do Amazon S3. Deve estar listado no grupo de parâmetros do cluster de banco de dados ou ocorrerá um erro.

Resposta

- id: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador exclusivo desse trabalho de processamento de dados.
- processingJob – Um objeto MlResourceDefinition.

  Definição do trabalho de processamento de dados.
- status: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O status do trabalho de processamento de dados.

Erros

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException

- TooManyRequestsException

CancelMLDataProcessingJob (ação)

O nome da CLI da AWS para essa API é: `cancel-ml-data-processing-job`.

Cancela um trabalho de processamento de dados do Neptune ML. Consulte O comando `dataprocessing`.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:CancelMLDataProcessingJob nesse cluster.

Solicitação

- clean (na CLI: `--clean`): um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se definido como `TRUE`, esse sinalizador especifica que todos os artefatos do S3 no Neptune ML deverão ser excluídos quando o trabalho for interrompido. O padrão é `FALSE`.
- id (na CLI: `--id`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador exclusivo do trabalho de processamento de dados.
- neptuneIamRoleArn (na CLI: `--neptune-iam-role-arn`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN de um perfil do IAM que concede ao Neptune acesso aos recursos do SageMaker e do Amazon S3. Deve estar listado no grupo de parâmetros do cluster de banco de dados ou ocorrerá um erro.

Resposta

- status: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O status da solicitação de cancelamento.

Erros

- UnsupportedOperationException

- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Estruturas de uso geral de ML:

MlResourceDefinition (estrutura)

Define um recurso do Neptune ML.

Campos

- arn: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN do recurso.
- cloudwatchLogUrl: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O URL de log do CloudWatch para o recurso.
- failureReason: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O motivo da falha, em caso de falha.
- name: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do recurso.
- outputLocation: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O local da saída.
- status: é uma string, do tipo: `string` (uma string codificada em UTF-8).

O status do recurso.

MlConfigDefinition (estrutura)

Contém uma configuração do Neptune ML.

Campos

- arn: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN da configuração.
- name: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome da configuração.

API de treinamento de modelos do Neptune ML

Ações de treinamento de modelos:

- StartMLModelTrainingJob (ação)
- ListMLModelTrainingJobs (ação)
- GetMLModelTrainingJob (ação)
- CancelMLModelTrainingJob (ação)

Estruturas de treinamento de modelos:

- CustomModelTrainingParameters (estrutura)

StartMLModelTrainingJob (ação)

O nome da CLI da AWS para essa API é: `start-ml-model-training-job`.

Cria um trabalho de treinamento de modelos do Neptune ML. Consulte Treinamento de modelos usando o comando `modeltraining`.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:StartMLModelTrainingJob nesse cluster.

Solicitação

- baseProcessingInstanceType (na CLI: `--base-processing-instance-type`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo de instância de ML usada na preparação e no gerenciamento do treinamento de modelos de ML. É uma instância de CPU escolhida com base nos requisitos de memória para processar os dados e o modelo de treinamento.

- customModelTrainingParameters (na CLI: `--custom-model-training-parameters`): um objeto CustomModelTrainingParameters.

  A configuração do treinamento de modelos personalizados. É um objeto JSON.

- dataProcessingJobId (na CLI: `--data-processing-job-id`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID do trabalho de processamento de dados concluído que criou os dados com os quais o treinamento funcionará.

- enableManagedSpotTraining (na CLI: `--enable-managed-spot-training`): um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Otimiza o custo de treinamento de modelos de machine learning usando instâncias spot do Amazon Elastic Compute Cloud. O padrão é `False`.

- id (na CLI: `--id`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um identificador exclusivo do novo trabalho. O padrão é um UUID gerado automaticamente.

- maxHPONumberOfTrainingJobs (na CLI: `--max-hpo-number-of-training-jobs`): um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Número total máximo de trabalhos de treinamento a serem iniciados para o trabalho de ajuste de hiperparâmetros. O padrão é 2. O Neptune ML ajusta automaticamente os hiperparâmetros do modelo de machine learning. Para obter um modelo com bom desempenho, use pelo menos dez trabalhos (em outras palavras, defina `maxHPONumberOfTrainingJobs` como dez). Em geral, quanto mais ajustes forem executados, melhores serão os resultados.

- maxHPOParallelTrainingJobs (na CLI: `--max-hpo-parallel-training-jobs`): um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Número máximo de trabalhos de treinamento paralelos a serem iniciados para o trabalho de ajuste de hiperparâmetros. O padrão é 2. O número de trabalhos paralelos que você pode executar é limitado pelos recursos disponíveis na instância de treinamento.

- neptuneIamRoleArn (na CLI: `--neptune-iam-role-arn`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN de um perfil do IAM que concede ao Neptune acesso aos recursos do SageMaker e do Amazon S3. Deve estar listado no grupo de parâmetros do cluster de banco de dados ou ocorrerá um erro.

- previousModelTrainingJobId (na CLI: `--previous-model-training-job-id`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID de um trabalho de treinamento de modelos concluído que você deseja atualizar de modo incremental com base nos dados atualizados.

- s3OutputEncryptionKMSKey (na CLI: `--s-3-output-encryption-kms-key`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  A chave do Amazon Key Management Service (KMS) que o SageMaker usa para criptografar a saída do trabalho de processamento. O padrão é nenhum.

- sagemakerIamRoleArn (na CLI: `--sagemaker-iam-role-arn`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN de um perfil do IAM para a execução do SageMaker. Ele deve estar listado no grupo de parâmetros do cluster de banco de dados ou ocorrerá um erro.

- securityGroupIds (na CLI: `--security-group-ids`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Os IDs dos grupos de segurança da VPC. O padrão é Nenhum.

- subnets (na CLI: `--subnets`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Os IDs das sub-redes na VPC do Neptune. O padrão é Nenhum.

- trainingInstanceType (na CLI: `--training-instance-type`): uma string, do tipo: `string` (uma string codificada em UTF-8).

O tipo de instância de ML usada para treinamento de modelos. Todos os modelos do Neptune ML são compatíveis com o treinamento de CPU, GPU e multiGPU. O padrão é `ml.p3.2xlarge`. A escolha do tipo de instância certo para treinamento depende do tipo de tarefa, do tamanho do grafo e do orçamento.

- trainingInstanceVolumeSizeInGB (na CLI: `--training-instance-volume-size-in-gb`): um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O tamanho do volume do disco da instância de treinamento. Tanto os dados de entrada quanto o modelo de saída são armazenados em disco, portanto, o tamanho do volume deve ser grande o suficiente para conter os dois conjuntos de dados. O padrão é 0. Se não for especificado ou for 0, o Neptune ML selecionará um tamanho de volume de disco com base na recomendação gerada na etapa de processamento de dados.
- trainingTimeOutInSeconds (na CLI: `--training-time-out-in-seconds`): um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  Tempo limite em segundos do trabalho de treinamento. O padrão é 86.400 (1 dia).
- trainModelS3Location (na CLI: `--train-model-s3-location`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O local no Amazon S3 onde os artefatos do modelo devem ser armazenados.
- volumeEncryptionKMSKey (na CLI: `--volume-encryption-kms-key`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  A chave do Amazon Key Management Service (KMS) que o SageMaker utiliza para criptografar dados no volume de armazenamento anexado às instâncias de computação de ML que executam o trabalho de treinamento. O padrão é Nenhum.

Resposta

- arn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN do novo trabalho de treinamento de modelos.
- creationTimeInMillis: é um valor longo, do tipo: `long` (um valor inteiro assinado de 64 bits).

  O tempo de criação do trabalho de treinamento do modelo, em milissegundos.
- id: uma string, do tipo: `string` (uma string codificada em UTF-8).

O ID exclusivo do novo trabalho de treinamento de modelos.

Erros

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

ListMLModelTrainingJobs (ação)

O nome da CLI da AWS para essa API é: `list-ml-model-training-jobs`.

Lista trabalhos de treinamento de modelos do Neptune ML. Consulte Treinamento de modelos usando o comando `modeltraining`.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:neptune-db:ListMLModelTrainingJobs nesse cluster.

Solicitação

- maxItems (na CLI :`--max-items`): um ListMLModelTrainingJobsInputMaxItemsInteger, do tipo: `integer` (um valor inteiro assinado de 32 bits), no mínimo 1 e até 1.024 ?st?s.

  O número máximo de itens a serem exibidos (de 1 a 1.024; o padrão é 10).
- neptuneIamRoleArn (na CLI: `--neptune-iam-role-arn`): uma string, do tipo: `string` (uma string codificada em UTF-8).

O ARN de um perfil do IAM que concede ao Neptune acesso aos recursos do SageMaker e do Amazon S3. Deve estar listado no grupo de parâmetros do cluster de banco de dados ou ocorrerá um erro.

Resposta

- ids: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma página da lista de IDs de trabalhos de treinamento de modelos.

Erros

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

GetMLModelTrainingJob (ação)

O nome da CLI da AWS para essa API é: `get-ml-model-training-job`.

Recupera informações sobre um trabalho de treinamento de modelos do Neptune ML. Consulte Treinamento de modelos usando o comando `modeltraining`.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:GetMLModelTrainingJobStatus nesse cluster.

Solicitação

- id (na CLI: `--id`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador exclusivo do trabalho de treinamento de modelos a ser recuperado.
- neptuneIamRoleArn (na CLI: `--neptune-iam-role-arn`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN de um perfil do IAM que concede ao Neptune acesso aos recursos do SageMaker e do Amazon S3. Deve estar listado no grupo de parâmetros do cluster de banco de dados ou ocorrerá um erro.

Resposta

- hpoJob – Um objeto MlResourceDefinition.

  O trabalho HPO.
- id: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador exclusivo do trabalho de treinamento de modelos.
- mlModels – Uma matriz de objetos MlConfigDefinition.

  Uma lista das configurações dos modelos de ML que estão sendo usados.
- modelTransformJob – Um objeto MlResourceDefinition.

  O trabalho de transformação de modelos.
- processingJob – Um objeto MlResourceDefinition.

  O trabalho do processamento de dados.
- status: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O status do novo trabalho de treinamento de modelos.

Erros

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException

- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

CancelMLModelTrainingJob (ação)

O nome da CLI da AWS para essa API é: `cancel-ml-model-training-job`.

Cancela um trabalho de treinamento de modelos do Neptune ML. Consulte Treinamento de modelos usando o comando `modeltraining`.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:CancelMLModelTrainingJob nesse cluster.

Solicitação

- clean (na CLI: `--clean`): um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se definido como `TRUE`, esse sinalizador especifica que todos os artefatos do Amazon S3 deverão ser excluídos quando o trabalho for interrompido. O padrão é `FALSE`.
- id (na CLI: `--id`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador exclusivo do trabalho de treinamento de modelos a ser cancelado.
- neptuneIamRoleArn (na CLI: `--neptune-iam-role-arn`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN de um perfil do IAM que concede ao Neptune acesso aos recursos do SageMaker e do Amazon S3. Deve estar listado no grupo de parâmetros do cluster de banco de dados ou ocorrerá um erro.

Resposta

- status: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O status do cancelamento.

Erros

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Estruturas de treinamento de modelos:

CustomModelTrainingParameters (estrutura)

Contém parâmetros de treinamento de modelos personalizados. Consulte Modelos personalizados no Neptune ML.

Campos

- sourceS3DirectoryPath: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O caminho para o local do Amazon S3 onde o módulo Python que implementa o modelo está localizado. Isso deve apontar para uma localização válida existente do Amazon S3 que contenha, no mínimo, um script de treinamento, um script de transformação e um arquivo `model-hpo-configuration.json`.

- trainingEntryPointScript: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do ponto de entrada no módulo de um script que executa o treinamento de modelos e usa hiperparâmetros como argumentos de linha de comando, incluindo hiperparâmetros fixos. O padrão é `training.py`.
- transformEntryPointScript: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do ponto de entrada no módulo de um script que deve ser executado após a identificação do melhor modelo da pesquisa de hiperparâmetros, para calcular os artefatos do modelo necessários para a implantação do modelo. Ele deve ser capaz de ser executado sem argumentos de linha de comando. O padrão é `transform.py`.

API de transformação de modelos do Neptune ML

Ações de transformação de modelos:

- StartMLModelTransformJob (ação)
- ListMLModelTransformJobs (ação)
- GetMLModelTransformJob (ação)
- CancelMLModelTransformJob (ação)

Estruturas de transformação de modelos:

- CustomModelTransformParameters (estrutura)

StartMLModelTransformJob (ação)

O nome da CLI da AWS para essa API é: `start-ml-model-transform-job`.

Cria um trabalho de transformação de modelos. Consulte Usar um modelo treinado para gerar novos artefatos de modelo.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:StartMLModelTransformJob nesse cluster.

Solicitação

- baseProcessingInstanceType (na CLI: `--base-processing-instance-type`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo de instância de ML usada na preparação e no gerenciamento do treinamento de modelos de ML. É uma instância de computação de ML escolhida com base nos requisitos de memória para processar os dados e o modelo de treinamento.

- baseProcessingInstanceVolumeSizeInGB (na CLI: `--base-processing-instance-volume-size-in-gb`): um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O tamanho do volume do disco da instância de treinamento em gigabytes. O padrão é 0. Tanto os dados de entrada quanto o modelo de saída são armazenados em disco, portanto, o tamanho do volume deve ser grande o suficiente para conter os dois conjuntos de dados. Se não for especificado ou for 0, o Neptune ML selecionará um tamanho de volume de disco com base na recomendação gerada na etapa de processamento de dados.

- customModelTransformParameters (na CLI: `--custom-model-transform-parameters`): um objeto CustomModelTransformParameters.

  Informações de configuração para uma transformação de modelos usando um modelo personalizado. O objeto `customModelTransformParameters` contém os seguintes campos, que devem ter valores compatíveis com os parâmetros do modelo salvos do trabalho de treinamento:

- dataProcessingJobId (na CLI: `--data-processing-job-id`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID de um trabalho de processamento de dados concluído. Você deve incluir `dataProcessingJobId` e especificar `mlModelTrainingJobId` ou `trainingJobName`.

- id (na CLI: `--id`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Um identificador exclusivo do novo trabalho. O padrão é um UUID gerado automaticamente.

- mlModelTrainingJobId (na CLI: `--ml-model-training-job-id`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID de um trabalho de treinamento de modelos concluído. Você deve incluir `dataProcessingJobId` e especificar `mlModelTrainingJobId` ou `trainingJobName`.

- modelTransformOutputS3Location (na CLI: `--model-transform-output-s3-location`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

O local no Amazon S3 onde os artefatos do modelo devem ser armazenados.

- neptuneIamRoleArn (na CLI: `--neptune-iam-role-arn`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN de um perfil do IAM que concede ao Neptune acesso aos recursos do SageMaker e do Amazon S3. Deve estar listado no grupo de parâmetros do cluster de banco de dados ou ocorrerá um erro.

- s3OutputEncryptionKMSKey (na CLI: `--s-3-output-encryption-kms-key`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  A chave do Amazon Key Management Service (KMS) que o SageMaker usa para criptografar a saída do trabalho de processamento. O padrão é nenhum.

- sagemakerIamRoleArn (na CLI: `--sagemaker-iam-role-arn`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN de um perfil do IAM para execução do SageMaker. Deve estar listado no grupo de parâmetros do cluster de banco de dados ou ocorrerá um erro.

- securityGroupIds (na CLI: `--security-group-ids`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Os IDs dos grupos de segurança da VPC. O padrão é Nenhum.

- subnets (na CLI: `--subnets`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Os IDs das sub-redes na VPC do Neptune. O padrão é Nenhum.

- trainingJobName (na CLI: `--training-job-name`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome de um trabalho de treinamento concluído do SageMaker. Você deve incluir `dataProcessingJobId` e especificar `mlModelTrainingJobId` ou `trainingJobName`.

- volumeEncryptionKMSKey (na CLI: `--volume-encryption-kms-key`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  A chave do Amazon Key Management Service (KMS) que o SageMaker utiliza para criptografar dados no volume de armazenamento anexado às instâncias de computação de ML que executam o trabalho de treinamento. O padrão é Nenhum.

Resposta

- arn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN do trabalho de treinamento de modelos.
- creationTimeInMillis: é um valor longo, do tipo: `long` (um valor inteiro assinado de 64 bits).

  O tempo de criação do trabalho de treinamento de modelos, em milissegundos.
- id: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID exclusivo do novo trabalho de treinamento de modelos.

Erros

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

ListMLModelTransformJobs (ação)

O nome da CLI da AWS para essa API é: `list-ml-model-transform-jobs`.

Exibe uma lista de IDs de trabalho de transformação de modelos. Consulte Usar um modelo treinado para gerar novos artefatos de modelo.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:ListMLModelTransformJobs nesse cluster.

Solicitação

- maxItems (na CLI :`--max-items`): um ListMLModelTransformJobsInputMaxItemsInteger, do tipo: `integer` (um valor inteiro assinado de 32 bits), no mínimo 1 e até 1.024 ?st?s.

  O número máximo de itens a serem exibidos (de 1 a 1.024; o padrão é 10).

- neptuneIamRoleArn (na CLI: `--neptune-iam-role-arn`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN de um perfil do IAM que concede ao Neptune acesso aos recursos do SageMaker e do Amazon S3. Deve estar listado no grupo de parâmetros do cluster de banco de dados ou ocorrerá um erro.

Resposta

- ids: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma página da lista de IDs de transformação de modelos.

Erros

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

GetMLModelTransformJob (ação)

O nome da CLI da AWS para essa API é: `get-ml-model-transform-job`.

Obtém informações sobre um trabalho de transformação de modelos específico. Consulte Usar um modelo treinado para gerar novos artefatos de modelo.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:GetMLModelTransformJobStatus nesse cluster.

Solicitação

- id (na CLI: `--id`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador exclusivo do trabalho de treinamento de modelos a ser recuperado.
- neptuneIamRoleArn (na CLI: `--neptune-iam-role-arn`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN de um perfil do IAM que concede ao Neptune acesso aos recursos do SageMaker e do Amazon S3. Deve estar listado no grupo de parâmetros do cluster de banco de dados ou ocorrerá um erro.

Resposta

- baseProcessingJob – Um objeto MlResourceDefinition.

  O trabalho de processamento de dados básico.
- id: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador exclusivo do trabalho de transformação de modelos a ser recuperado.
- models – Uma matriz de objetos MlConfigDefinition.

  Uma lista das informações de configurações dos modelos de ML utilizados.
- remoteModelTransformJob – Um objeto MlResourceDefinition.

  O trabalho de transformação de modelos remotos.
- status: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O status do trabalho de transformação de modelos.

Erros

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

CancelMLModelTransformJob (ação)

O nome da CLI da AWS para essa API é: `cancel-ml-model-transform-job`.

Cancela um trabalho de transformação de modelos especificado. Consulte Usar um modelo treinado para gerar novos artefatos de modelo.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:CancelMLModelTransformJob nesse cluster.

Solicitação

- clean (na CLI: `--clean`): um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se esse sinalizador for definido como `TRUE`, todos os artefatos do S3 no Neptune ML deverão ser excluídos quando o trabalho for interrompido. O padrão é `FALSE`.
- id (na CLI: `--id`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador exclusivo do trabalho de transformação de modelos a ser cancelado.
- neptuneIamRoleArn (na CLI: `--neptune-iam-role-arn`): uma string, do tipo: `string` (uma string codificada em UTF-8).

O ARN de um perfil do IAM que concede ao Neptune acesso aos recursos do SageMaker e do Amazon S3. Deve estar listado no grupo de parâmetros do cluster de banco de dados ou ocorrerá um erro.

Resposta

- status: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O status do cancelamento.

Erros

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Estruturas de transformação de modelos:

CustomModelTransformParameters (estrutura)

Contém parâmetros de transformação de modelos personalizados. Consulte Usar um modelo treinado para gerar novos artefatos de modelo.

Campos

- sourceS3DirectoryPath: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

O caminho para o local do Amazon S3 onde o módulo Python que implementa o modelo está localizado. Isso deve apontar para uma localização válida existente do Amazon S3 que contenha, no mínimo, um script de treinamento, um script de transformação e um arquivo `model-hpo-configuration.json`.

- transformEntryPointScript: é uma string, do tipo: `string` (uma string codificada em UTF-8).

  O nome do ponto de entrada no módulo de um script que deve ser executado após a identificação do melhor modelo da pesquisa de hiperparâmetros, para calcular os artefatos do modelo necessários para a implantação do modelo. Ele deve ser capaz de ser executado sem argumentos de linha de comando. O padrão é `transform.py`.

API de endpoint de inferência do Neptune ML

Ações de endpoint de inferência:

- CreateMLEndpoint (ação)
- ListMLEndpoints (ação)
- GetMLEndpoint (ação)
- DeleteMLEndpoint (ação)

CreateMLEndpoint (ação)

O nome da CLI da AWS para essa API é: `create-ml-endpoint`.

Cria um endpoint de inferência do Neptune ML que permite consultar um modelo específico que o processo de treinamento de modelos construiu. Consulte Gerenciar endpoints de inferência usando o comando.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:CreateMLEndpoint nesse cluster.

Solicitação

- id (na CLI: `--id`): uma string, do tipo: `string` (uma string codificada em UTF-8).

Um identificador exclusivo para o novo endpoint de inferência. O padrão é um nome com carimbo de data e hora gerado automaticamente.

- instanceCount (na CLI: `--instance-count`): um valor inteiro, do tipo: `integer` (um valor inteiro assinado de 32 bits).

  O número mínimo de instâncias do Amazon EC2 a serem implantadas em um endpoint para previsão. O padrão é 1.

- instanceType (na CLI: `--instance-type`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O tipo de instância do Neptune ML usada para serviços on-line. O padrão é `ml.m5.xlarge`. Selecionar a instância de ML para um endpoint de inferência depende do tipo de tarefa, do tamanho do grafo e do orçamento.

- mlModelTrainingJobId (na CLI: `--ml-model-training-job-id`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID do trabalho de treinamento de modelos concluído que criou o modelo para o qual o endpoint de inferência apontará. É necessário fornecer o `mlModelTrainingJobId` ou o `mlModelTransformJobId`.

- mlModelTransformJobId (na CLI: `--ml-model-transform-job-id`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID do trabalho de transformação de modelos concluído. É necessário fornecer o `mlModelTrainingJobId` ou o `mlModelTransformJobId`.

- modelName (na CLI: `--model-name`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  Tipo de modelo para treinamento. Por padrão, o modelo do Neptune ML é automaticamente baseado no `modelType` usado no processamento de dados, mas você pode especificar um tipo de modelo diferente aqui. O padrão é `rgcn` para grafos heterogêneos e `kge` para grafos de conhecimento. O único valor válido para grafos heterogêneos é `rgcn`. Os valores válidos para grafos de conhecimento são: `kge`, `transe`, `distmult` e `rotate`.

- neptuneIamRoleArn (na CLI: `--neptune-iam-role-arn`): uma string, do tipo: `string` (uma string codificada em UTF-8).

O ARN de um perfil do IAM que concede ao Neptune acesso aos recursos do SageMaker e do Amazon S3. Deve estar listado no grupo de parâmetros do cluster de banco de dados ou será lançado um erro.

- update (na CLI: `--update`): um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se definido como `true`, `update` indicará que se trata de uma solicitação de atualização. O padrão é `false`. É necessário fornecer o `mlModelTrainingJobId` ou o `mlModelTransformJobId`.

- volumeEncryptionKMSKey (na CLI: `--volume-encryption-kms-key`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  A chave do Amazon Key Management Service (Amazon KMS) que o SageMaker utiliza para criptografar dados no volume de armazenamento anexado às instâncias de computação de ML que executam o trabalho de treinamento. O padrão é Nenhum.

Resposta

- arn: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN do novo endpoint de inferência.

- creationTimeInMillis: é um valor longo, do tipo: `long` (um valor inteiro assinado de 64 bits).

  O tempo de criação do endpoint, em milissegundos.

- id: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID exclusivo do novo endpoint de inferência.

Erros

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException

- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

ListMLEndpoints (ação)

O nome da CLI da AWS para essa API é: `list-ml-endpoints`.

Lista endpoints de inferência existentes. Consulte Gerenciar endpoints de inferência usando o comando.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:ListMLEndpoints nesse cluster.

Solicitação

- maxItems (na CLI :`--max-items`): um ListMLEndpointsInputMaxItemsInteger, do tipo: `integer` (um valor inteiro assinado de 32 bits), no mínimo 1 e até 1.024 ?st?s.

  O número máximo de itens a serem exibidos (de 1 a 1.024; o padrão é 10).
- neptuneIamRoleArn (na CLI: `--neptune-iam-role-arn`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN de um perfil do IAM que concede ao Neptune acesso aos recursos do SageMaker e do Amazon S3. Deve estar listado no grupo de parâmetros do cluster de banco de dados ou ocorrerá um erro.

Resposta

- ids: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma página da lista de IDs de endpoints de inferência.

Erros

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

GetMLEndpoint (ação)

O nome da CLI da AWS para essa API é: `get-ml-endpoint`.

Recupera detalhes sobre um endpoint de inferência. Consulte Gerenciar endpoints de inferência usando o comando.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:GetMLEndpointStatus nesse cluster.

Solicitação

- id (na CLI: `--id`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador exclusivo do endpoint de inferência.
- neptuneIamRoleArn (na CLI: `--neptune-iam-role-arn`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN de um perfil do IAM que concede ao Neptune acesso aos recursos do SageMaker e do Amazon S3. Deve estar listado no grupo de parâmetros do cluster de banco de dados ou ocorrerá um erro.

Resposta

- endpoint – Um objeto MlResourceDefinition.

  A definição de endpoint.
- endpointConfig – Um objeto MlConfigDefinition.

  A configuração de endpoint.
- id: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador exclusivo do endpoint de inferência.
- status: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O status do endpoint de inferência.

Erros

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

DeleteMLEndpoint (ação)

O nome da CLI da AWS para essa API é: `delete-ml-endpoint`.

Cancela a criação de um endpoint de inferência do Neptune ML. Consulte Gerenciar endpoints de inferência usando o comando.

Ao invocar essa operação em um cluster do Neptune com a autenticação do IAM habilitada, o usuário ou o perfil do IAM que faz a solicitação deve ter uma política anexada que viabilize a ação do IAM neptune-db:DeleteMLEndpoint nesse cluster.

Solicitação

- clean (na CLI: `--clean`): um valor booliano, do tipo: `boolean` (um valor booliano (verdadeiro ou falso)).

  Se esse sinalizador for definido como `TRUE`, todos os artefatos do S3 no Neptune ML deverão ser excluídos quando o trabalho for interrompido. O padrão é `FALSE`.
- id (na CLI: `--id`): obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O identificador exclusivo do endpoint de inferência.
- neptuneIamRoleArn (na CLI: `--neptune-iam-role-arn`): uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ARN de um perfil do IAM que concede ao Neptune acesso aos recursos do SageMaker e do Amazon S3. Deve estar listado no grupo de parâmetros do cluster de banco de dados ou será lançado um erro.

Resposta

- status: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O status do cancelamento.

Erros

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException

- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Exceções da API do plano de dados do Neptune

Exceções:

- AccessDeniedException (estrutura)
- BadRequestException (estrutura)
- BulkLoadIdNotFoundException (estrutura)
- CancelledByUserException (estrutura)
- ClientTimeoutException (estrutura)
- ConcurrentModificationException (estrutura)
- ConstraintViolationException (estrutura)
- ExpiredStreamException (estrutura)
- FailureByQueryException (estrutura)
- IllegalArgumentException (estrutura)
- InternalFailureException (estrutura)
- InvalidArgumentException (estrutura)
- InvalidNumericDataException (estrutura)
- InvalidParameterException (estrutura)
- LoadUrlAccessDeniedException (estrutura)
- MalformedQueryException (estrutura)
- MemoryLimitExceededException (estrutura)
- MethodNotAllowedException (estrutura)
- MissingParameterException (estrutura)
- MLResourceNotFoundException (estrutura)
- ParsingException (estrutura)
- PreconditionsFailedException (estrutura)
- QueryLimitExceededException (estrutura)

- QueryLimitException (estrutura)
- QueryTooLargeException (estrutura)
- ReadOnlyViolationException (estrutura)
- S3Exception (estrutura)
- ServerShutdownException (estrutura)
- StatisticsNotAvailableException (estrutura)
- StreamRecordsNotFoundException (estrutura)
- ThrottlingException (estrutura)
- TimeLimitExceededException (estrutura)
- TooManyRequestsException (estrutura)
- UnsupportedOperationException (estrutura)
- UnloadUrlAccessDeniedException (estrutura)

AccessDeniedException (estrutura)

Gerada em caso de falha na autenticação ou na autorização.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.
- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da execução em questão.

BadRequestException (estrutura)

Gerada quando uma solicitação é enviada e não pode ser processada.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

Código de status HTTP gerado com a exceção.

- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.
- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da solicitação inadequada.

BulkLoadIdNotFoundException (estrutura)

Gerada quando um ID de trabalho de carregamento em massa especificado não pode ser encontrado.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.
- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID do trabalho de carregamento em massa que não pôde ser encontrado.

CancelledByUserException (estrutura)

Gerada quando um usuário cancelou uma solicitação.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.
- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

O ID da execução em questão.

ClientTimeoutException (estrutura)

Gerada quando uma solicitação atingiu o tempo limite no cliente.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.
- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da execução em questão.

ConcurrentModificationException (estrutura)

Gerada quando uma solicitação tenta modificar dados que estão sendo modificados simultaneamente por outro processo.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.
- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da execução em questão.

ConstraintViolationException (estrutura)

Gerada quando um valor em um campo de solicitação não satisfaz as restrições exigidas.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.
- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da execução em questão.

ExpiredStreamException (estrutura)

Gerada quando uma solicitação tenta acessar um fluxo que expirou.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.
- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da execução em questão.

FailureByQueryException (estrutura)

Gerada quando há falha em uma solicitação.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.

- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da execução em questão.

IllegalArgumentException (estrutura)

Gerada quando um argumento em uma solicitação não é compatível.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.
- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da execução em questão.

InternalFailureException (estrutura)

Gerada quando o processamento da solicitação falhou inesperadamente.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.
- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da execução em questão.

InvalidArgumentException (estrutura)

Gerada quando um argumento em uma solicitação tem um valor inválido.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.
- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da execução em questão.

InvalidNumericDataException (estrutura)

Gerada quando dados numéricos inválidos são encontrados ao atender a uma solicitação.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.
- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da execução em questão.

InvalidParameterException (estrutura)

Gerada quando o valor de um parâmetro não é válido.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.

- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da solicitação que inclui um parâmetro inválido.

LoadUrlAccessDeniedException (estrutura)

Gerada quando o acesso é negado a um URL de carregamento especificado.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.
- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da execução em questão.

MalformedQueryException (estrutura)

Gerada quando uma consulta é enviada que está sintaticamente incorreta ou não passa pela validação adicional.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.
- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da solicitação de consulta malformada.

MemoryLimitExceededException (estrutura)

Gerada quando uma solicitação falha devido a recursos de memória insuficientes. A solicitação pode ser repetida.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.
- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da solicitação que falhou.

MethodNotAllowedException (estrutura)

Gerada quando o método HTTP usado por uma solicitação não é compatível com o endpoint que está sendo usado.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.
- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da execução em questão.

MissingParameterException (estrutura)

Gerada quando falta um parâmetro necessário.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.
- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da solicitação na qual o parâmetro está ausente.

MLResourceNotFoundException (estrutura)

Gerada quando um recurso de machine learning especificado não pôde ser encontrado.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.
- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da execução em questão.

ParsingException (estrutura)

Gerada quando um problema de análise é encontrado.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.

- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da execução em questão.

PreconditionsFailedException (estrutura)

Gerada quando uma condição prévia para o processamento de uma solicitação não é satisfeita.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.
- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da execução em questão.

QueryLimitExceededException (estrutura)

Gerada quando o número de consultas ativas excede o que o servidor pode processar. A consulta em questão poderá ser repetida quando o sistema estiver menos ocupado.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.
- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da solicitação que excedeu o limite.

QueryLimitException (estrutura)

Gerada quando o tamanho de uma consulta excede o limite do sistema.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.
- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da solicitação que excedeu o limite.

QueryTooLargeException (estrutura)

Gerada quando o corpo de uma consulta é grande demais.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.
- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da solicitação que é grande demais.

ReadOnlyViolationException (estrutura)

Gerada quando uma solicitação tenta gravar em um recurso somente leitura.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

Código de status HTTP gerado com a exceção.

- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.
- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da solicitação na qual o parâmetro está ausente.

S3Exception (estrutura)

Gerada quando há um problema ao acessar o Amazon S3.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.
- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da execução em questão.

ServerShutdownException (estrutura)

Gerada quando o servidor é desligado durante o processamento de uma solicitação.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.
- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

O ID da execução em questão.

StatisticsNotAvailableException (estrutura)

Gerada quando as estatísticas necessárias para atender a uma solicitação não estão disponíveis.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.
- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da execução em questão.

StreamRecordsNotFoundException (estrutura)

Gerada quando os registros de fluxo solicitados por uma consulta não podem ser encontrados.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.
- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da execução em questão.

ThrottlingException (estrutura)

Gerada quando a taxa de solicitações excede o throughput máximo. As solicitações podem ser repetidas depois de encontrar essa exceção.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.
- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da solicitação que não pôde ser processada por esse motivo.

TimeLimitExceededException (estrutura)

Gerada quando a operação excede o limite de tempo permitido para ela.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.
- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da solicitação que não pôde ser processada por esse motivo.

TooManyRequestsException (estrutura)

Gerada quando o número de solicitações processadas excede o limite.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.

- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da solicitação que não pôde ser processada por esse motivo.

UnsupportedOperationException (estrutura)

Gerada quando uma solicitação tenta iniciar uma operação que não é compatível.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.
- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da execução em questão.

UnloadUrlAccessDeniedException (estrutura)

Gerada quando o acesso é negado a um URL que é um destino de descarregamento.

Campos

- code: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Código de status HTTP gerado com a exceção.
- detailedMessage: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  Uma mensagem detalhada que descreve o problema.
- requestId: é obrigatório: uma string, do tipo: `string` (uma string codificada em UTF-8).

  O ID da execução em questão.

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.