aws

Guide du développeur

Amazon Quantum Ledger Database (Amazon QLDB)

Amazon Quantum Ledger Database (Amazon QLDB): Guide du développeur

Table of Contents

Présentation d'Amazon QLDB

La base de données Amazon Quantum Ledger (Amazon QLDB) est une base de données de registre entièrement gérée qui fournit un journal des transactions transparent, immuable et vérifiable par cryptographie, appartenant à une autorité centrale de confiance. Vous pouvez utiliser Amazon QLDB pour suivre chaque modification de données d'application et maintenir un historique complet et vérifiable des modifications au fil du temps. Pour en savoir plus sur les différentes options de base de données disponibles sur Amazon Web Services, consultez la section Choisir la base de données adaptée à votre organisation surAWS.

Les registres sont généralement utilisés pour enregistrer l'historique des activités économiques et financières d'une organisation. De nombreuses entreprises créent des applications dotées de fonctionnalités similaires à celles d'un registre, car elles souhaitent conserver un historique précis des données de leurs applications. Par exemple, ils peuvent souhaiter suivre l'historique des crédits et des débits dans le cadre de transactions bancaires, vérifier la traçabilité des données d'une réclamation d'assurance ou suivre le mouvement d'un article dans un réseau de chaîne d'approvisionnement. Les applications Ledger sont souvent mises en œuvre à l'aide de tables d'audit personnalisées ou de pistes d'audit créées dans des bases de données relationnelles.

Amazon QLDB est une nouvelle catégorie de base de données qui permet d'éliminer la nécessité de participer à l'effort de développement complexe lié à la création de vos propres applications de type registre. Avec QLDB, l'historique des modifications apportées à vos données est immuable : il ne peut pas être remplacé ou modifié sur place. Et grâce à la cryptographie, vous pouvez vérifier qu'aucune modification involontaire n'a été apportée aux données de votre application. QLDB utilise un journal transactionnel immuable, appelé journal. Le journal est uniquement accessible en annexe et se compose d'un ensemble de blocs séquencés et enchaînés par hachage qui contiennent vos données validées.

Vidéo Amazon QLDB

Pour une présentation d'Amazon QLDB et des avantages qu'il peut vous apporter, regardez cette vidéo de présentation de QLDB sur YouTube.

Tarification Amazon QLDB

Avec Amazon QLDB, vous ne payez que pour ce que vous utilisez et il n'existe pas de frais d'utilisation ou de service obligatoire. Vous ne payez que pour les ressources consommées par votre base de données, et vous n'avez pas besoin de les provisionner à l'avance.

Pour de plus amples informations, veuillez consulter Tarification Amazon QLDB.

Démarrage avec QLDB

Nous vous recommandons de commencer par lire les rubriques suivantes :

- Présentation d'Amazon QLDB— Pour une présentation générale de QLDB.
- Concepts et terminologie de base d'Amazon QLDB— Pour apprendre les concepts et la terminologie fondamentaux des QLDB.
- Accès à Amazon QLDB— Pour savoir comment accéder à QLDB à l'aide de l'APIAWS Management Console, ouAWS Command Line Interface (AWS CLI).
- Comment Amazon QLDB fonctionne avec IAM— Pour apprendre à contrôler l'accès à QLDB à l'aide deAWS Identity and Access Management (IAM).

Pour commencer rapidement à utiliser la console QLDB, consultezMise en route avec la console Amazon QLDB.

Pour en savoir plus sur le développement avec QLDB à l'aide d'un piloteAWS fourni, consultezDémarrage QLDB.

Présentation d'Amazon QLDB

Les sections suivantes présentent de haut niveau les composants du service Amazon QLDB et leurs interactions.

Rubriques

- Journal d'abord
- Immuable
- Vérifiable par cryptographie
- Similaire à SQL et flexible en matière de documents

- Outils de développement open source
- Sans serveur et haute disponibilité
- Niveau Enterprise

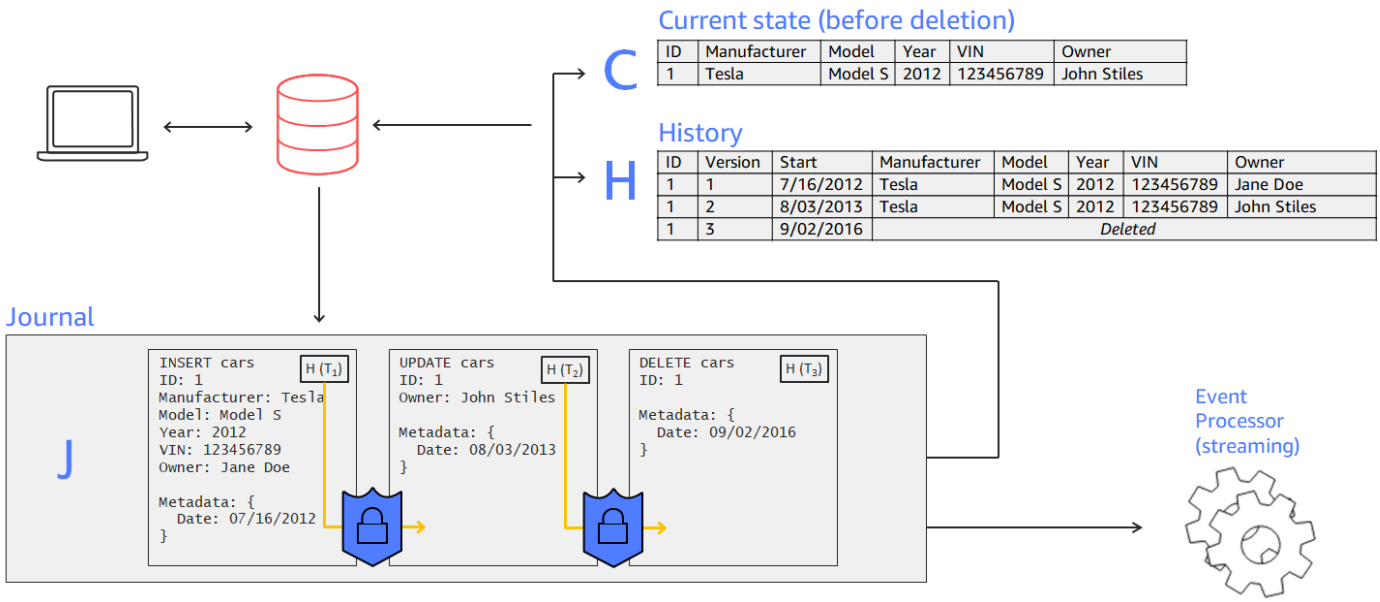
Journal d'abord

Dans l'architecture de base de données traditionnelle, vous écrivez généralement des données dans des tables dans le cadre d'une transaction. Un journal des transactions, généralement une implémentation interne, enregistre toutes les transactions et les modifications de base de données qu'elles apportent. Le journal des transactions est un élément essentiel de la base de données. Vous avez besoin du journal pour rejouer les transactions en cas de panne du système, de reprise après sinistre ou de réplication de données. Toutefois, les journaux de transactions des bases de données ne sont pas immuables et ne sont pas conçus pour fournir un accès direct et facile aux utilisateurs.

Dans Amazon QLDB, le journal constitue le cœur de la base de données. Structurellement similaire à un journal de transactions, le journal est une structure de données immuable, à ajout uniquement, qui stocke les données de votre application ainsi que les métadonnées associées. Toutes les transactions d'écriture, y compris les mises à jour et les suppressions, sont d'abord validées dans le journal.

QLDB utilise le journal pour déterminer l'état actuel des données de votre registre en les matérialisant dans des tables interrogeables définies par l'utilisateur. Ces tableaux fournissent également un historique accessible de toutes les données de transaction, y compris les révisions des documents et les métadonnées. En outre, le journal gère la simultanéité, le séquençage, la vérification cryptographique et la disponibilité des données du registre.

Le diagramme suivant illustre l'architecture du journal QLDB.



- Dans cet exemple, une application se connecte à un registre et exécute des transactions qui insèrent, mettent à jour et suppriment un document dans une table nommée`cars`.
- Les données sont d'abord écrites dans le journal dans un ordre séquentiel.
- Les données sont ensuite matérialisées dans le tableau avec des vues intégrées. Ces vues vous permettent de consulter à la fois l'état actuel et l'historique complet de la voiture, un numéro de version étant attribué à chaque révision.
- Vous pouvez également exporter ou diffuser des données directement depuis le journal.

Immuable

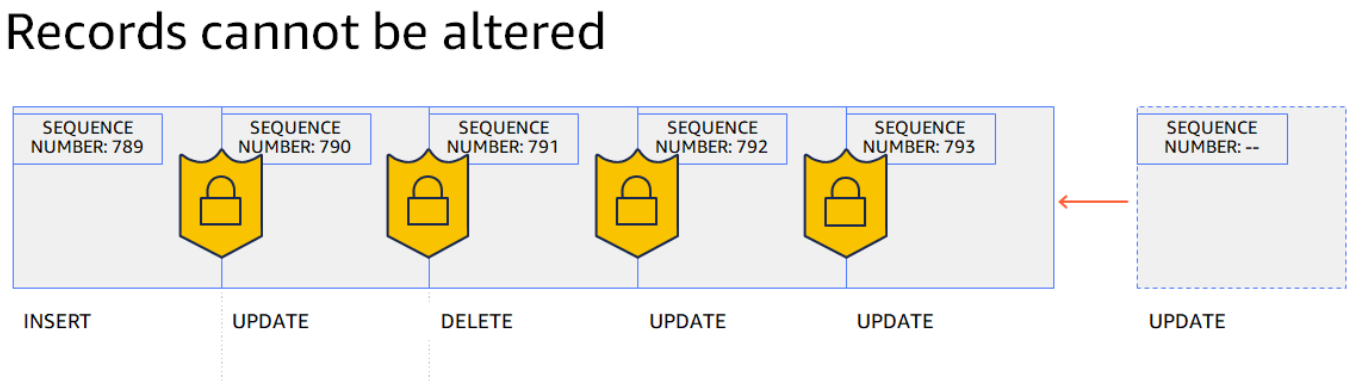
Étant donné que le journal QLDB est uniquement un ajout, il conserve un enregistrement complet de toutes les modifications apportées à vos données qui ne peuvent être ni modifiées ni remplacées. Il n'existe aucune API ou autre méthode permettant de modifier les données validées en place. Cette structure de journal vous permet d'accéder à l'historique complet de votre registre et de l'interroger.

> **Note**
>
> La seule exception à l'immuabilité prise en charge par QLDB est la rédaction de données. Grâce à cette fonctionnalité, vous pouvez vous conformer aux lois réglementaires telles que

le règlement général sur la protection des données (RGPD) de l'Union européenne et la loi californienne sur la protection de la vie privée des consommateurs (CCPA).
QLDB fournit une opération de rédaction qui vous permet de supprimer définitivement les révisions de documents inactives dans l'historique d'un tableau. Cette opération supprime uniquement les données utilisateur de la révision spécifiée et laisse la séquence du journal et les métadonnées du document inchangées. Cela permet de préserver l'intégrité globale des données de votre registre. Pour plus d'informations, veuillez consulter Rédaction de révisions de documents.

QLDB écrit un bloc dans le journal lors d'une transaction. Chaque bloc contient des objets d'entrée qui représentent les documents que vous insérez, mettez à jour et supprimez, ainsi que les instructions que vous avez exécutées pour les valider. Ces blocs sont séquencés et enchaînés par hachage pour garantir l'intégrité des données.

Le diagramme suivant illustre la structure de ce journal.



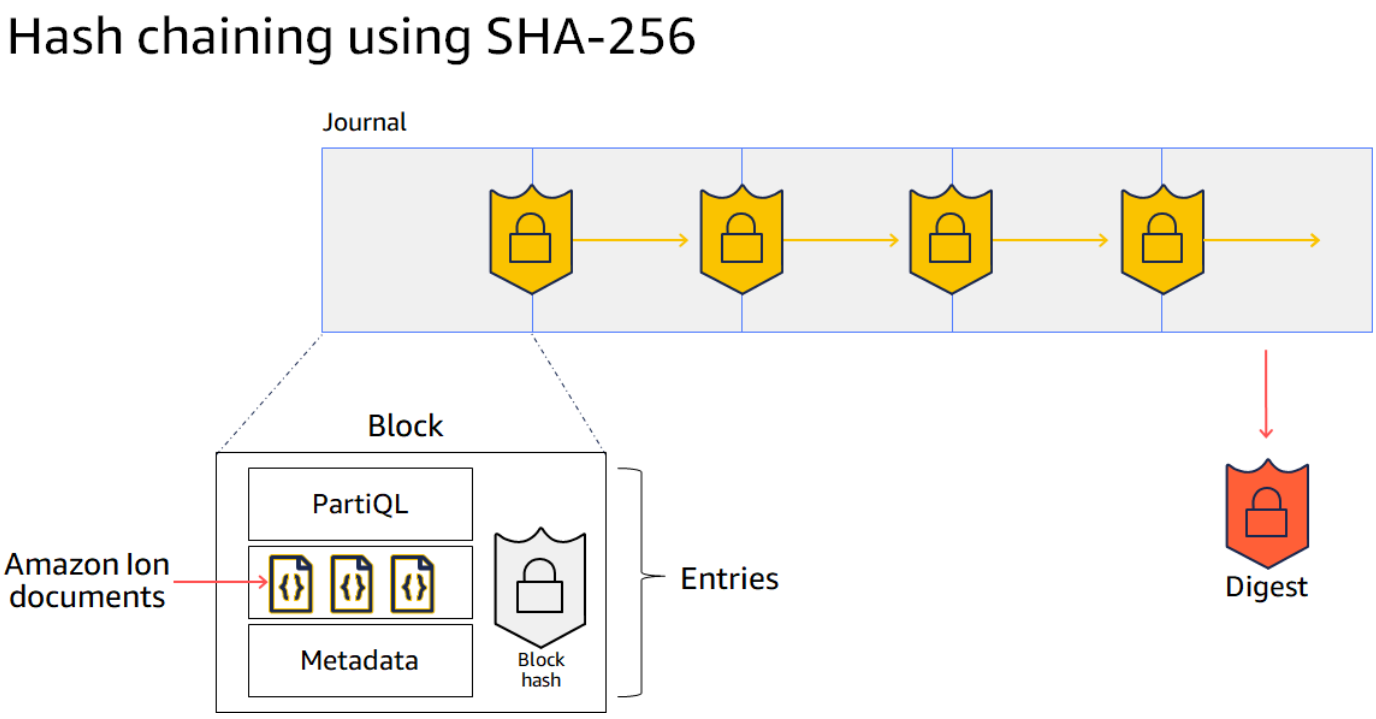
Le diagramme montre que les transactions sont enregistrées dans le journal sous forme de blocs qui sont enchaînés par hachage à des fins de vérification. Chaque bloc possède un numéro de séquence qui indique son adresse.

Vérifiable par cryptographie

Les blocs de journal sont séquencés et enchaînés à l'aide de techniques de hachage cryptographique, similaires aux blockchains. QLDB utilise la chaîne de hachage de la revue pour garantir l'intégrité des données transactionnelles à l'aide d'une méthode de vérification cryptographique. À l'aide d'un condensé (une valeur de hachage qui représente la chaîne de hachage complète d'une revue à un moment donné) et d'une preuve d'audit Merkle (un mécanisme qui prouve

la validité de n'importe quel nœud d'un arbre de hachage binaire), vous pouvez vérifier à tout moment qu'aucune modification involontaire n'a été apportée à vos données.

Le schéma suivant montre un condensé qui couvre la chaîne de hachage complète d'une revue à un moment donné.



Dans ce diagramme, les blocs de journal sont hachés à l'aide de la fonction de hachage cryptographique SHA-256 et sont enchaînés séquentiellement aux blocs suivants. Chaque bloc contient des entrées qui incluent vos documents de données, vos métadonnées et les instructions PartiQL exécutées lors de la transaction.

Pour plus d'informations, veuillez consulter Vérification des données dans Amazon QLDB.

Similaire à SQL et flexible en matière de documents

QLDB utilise PartiQL comme langage de requête et Amazon Ion comme modèle de données orienté document. PartiQL est un langage de requête open source compatible SQL qui a été étendu pour fonctionner avec Ion. Avec PartiQL, vous pouvez insérer, interroger et gérer vos données à l'aide d'opérateurs SQL familiers. Lorsque vous interrogez des documents plats, la syntaxe est la même que lorsque vous utilisez SQL pour interroger des tables relationnelles. Pour en savoir plus sur l'implémentation QLDB de PartiQL, consultez leRéférence Amazon QLDB.

Amazon Ion est un sur-ensemble de JSON. Ion est un format de données open source basé sur des documents qui vous permet de stocker et de traiter des données structurées, semi-structurées et imbriquées. Pour en savoir plus sur Ion dans QLDB, consultez leRéférence du format de données Amazon Ion dans Amazon QLDB.

Pour une comparaison approfondie des principaux composants et fonctionnalités des bases de données relationnelles traditionnelles par rapport à QLDB, consultezDu relationnel au grand livre.

Outils de développement open source

Pour simplifier le développement d'applications, QLDB fournit des pilotes open source dans différents langages de programmation. Vous pouvez utiliser ces pilotes pour interagir avec l'API de données transactionnelles en exécutant des instructions PartiQL sur un registre et en traitant les résultats de ces instructions. Pour obtenir des informations et des didacticiels sur les langues de pilote actuellement prises en charge, consultezDémarrage QLDB.

Amazon Ion fournit également des bibliothèques clientes qui traitent les données Ion pour vous. Pour consulter des guides pour les développeurs et des exemples de code relatifs au traitement des données Ion, consultez la documentation Amazon Ion sur GitHub.

Sans serveur et haute disponibilité

QLDB est entièrement géré, sans serveur et hautement disponible. Le service évolue automatiquement pour répondre aux exigences de votre application, et vous n'avez pas besoin de provisionner des instances ou de la capacité. Plusieurs copies de vos données sont répliquées au sein d'une zone de disponibilité et entre les zones de disponibilité d'uneRégion AWS.

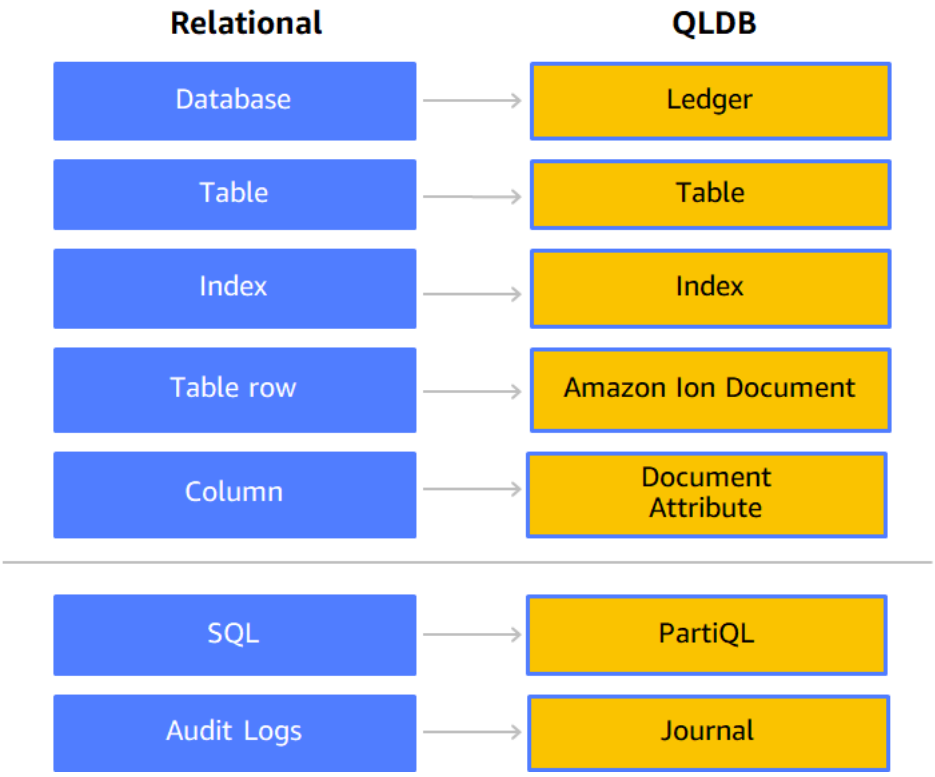
Niveau Enterprise

Les transactions QLDB sont entièrement conformes aux propriétés d'atomicité, de cohérence, d'isolation et de durabilité (ACID). QLDB utilise un contrôle de simultanéité optimiste (OCC) et les transactions fonctionnent avec une sérialisation complète, soit le plus haut niveau d'isolation. Cela signifie qu'il n'y a aucun risque de voir des lectures fantômes, des lectures erronées, des erreurs d'écriture ou d'autres problèmes de simultanéité similaires. Pour plus d'informations, veuillez consulter Modèle de mise QLDB simultansimultansimultansimultansimultansimultansimultansimultansimultanéité.

Du relationnel au grand livre

Si vous êtes développeur d'applications, il se peut que vous ayez une certaine expérience concernant l'utilisation d'un système de gestion de base de données relationnelle (SGBDR) et d'un langage de recherche structurée (SQL). Lorsque vous commencerez à utiliser Amazon QLDB, vous constaterez qu'il présente bon nombre de similitudes. Au fur et à mesure que vous progressez vers des sujets plus avancés, vous découvrirez également de nouvelles fonctionnalités puissantes que QLDB a développées sur la base du SGBDR. Cette section décrit les composants et les opérations courants de base de données, en les comparant et leurs équivalents dans QLDB.

Le schéma suivant montre les structures de mappage des composants principaux entre un SGBDR traditionnel et Amazon QLDB.



Le tableau suivant présente les principales similitudes et différences de haut niveau des fonctionnalités opérationnelles intégrées entre un SGBDR traditionnel et un QLDB.

Opération	SGBDR	QLDB
Création de tables	`CREATE TABLE` instruction qui définit tous les noms des	`CREATE TABLE` déclaration qui ne définit aucun attribut

Opération	SGBDR	QLDB
	colonnes et les types de données	de table ni aucun type de données afin d'autoriser un contenu ouvert et sans schéma
Création d'index	`CREATE INDEX` déclaration	`CREATE INDEX`instruction pour tous les champs de niveau supérieur d'une table
Insertion de données	`INSERT`instruction qui spécifie des valeurs au sein d'une nouvelle ligne ou d'un nouveau tuple qui adhère au schéma tel que défini par la table	`INSERT`instruction qui spécifie les valeurs d'un nouveau document dans n'importe quel format Amazon Ion valide, quels que soient les documents existants dans le tableau
Interrogation de données	`SELECT-FROM-WHERE` déclaration	`SELECT-FROM-WHERE`instruction utilisant la même syntaxe que SQL lors de l'interrogation de documents plats
Mise à jour de données	`UPDATE-SET-WHERE` déclaration	`UPDATE-SET-WHERE` instruction dans la même syntaxe que SQL lors de la mise à jour de documents plats
Suppression des données	`DELETE-FROM-WHERE` déclaration	`DELETE-FROM-WHERE`instruction dans la même syntaxe que SQL lors de la suppression de documents plats

Opération	SGBDR	QLDB
Données imbriquées et semi-structurées	Lignes plates ou tuples uniquement	Documents pouvant contenir n'importe quelles données structurées, semi-structurées ou imbriquées, conformément au format de données Amazon Ion et au langage de requête PartiQL
Interrogation de métadonnées	Aucune métadonnée intégrée	`SELECT`instruction qui interroge à partir de la vue validée intégrée d'une table
Consultation de l'historique des révisions	Pas d'historique des données intégré	`SELECT`instruction qui interroge à partir de la fonction d'historique intégrée
Vérification cryptographique	Aucune cryptographie ou immuabilité intégrées	API qui renvoient le résumé d'un journal et une preuve vérifiant l'intégrité de toute révision de document par rapport à ce condensé

Pour une présentation des concepts de base et de la terminologie de QLDB, consultezConcepts de base.

Pour plus d'informations sur le processus de création, d'interrogation et de gestion de vos données dans un registre, consultezUtilisation des données et de l'historique.

Concepts et terminologie de base d'Amazon QLDB

Cette section fournit une vue d'ensemble des concepts et de la terminologie de base d'Amazon QLDB, notamment la structure du registre et la façon dont un registre gère les données. En tant que base de données de registre, QLDB se distingue des autres bases de données orientées documents en ce qui concerne les concepts clés suivants.

Rubriques

Modèle d'objet de données QLDB

Le modèle d'objet de données fondamental dans Amazon QLDB est décrit comme suit :

1. Grand livre

   La première étape consiste à créer un registre, qui est le type deAWS ressource principal dans QLDB. Pour savoir comment créer un registre, consultez la section PriseÉtape 1 : Créer un nouveau registre en main de la console ouOpérations de base pour les registres Amazon QLDB.

   Pour les modes d'`STANDARD`autorisation`ALLOW_ALL` et les modes d'autorisation d'un registre, vous créez des politiquesAWS Identity and Access Management (IAM) qui accordent des autorisations pour exécuter des opérations d'API sur cette ressource de registre.

   Format ARN de transaction :

   ```
   arn:aws:qldb:${region}:${account-id}:ledger/${ledger-name}
   ```

2. Journal et tableaux

   Pour commencer à écrire des données dans un registre QLDB, vous devez d'abord créer une table avec uneCREATE TABLE instruction de base. Les données du grand livre sont des révisions de documents qui sont enregistrées dans le journal du grand livre. Vous enregistrez les révisions des documents dans le grand livre dans le contexte de tables définies par l'utilisateur. Dans QLDB, un tableau représente une vue matérialisée d'un ensemble de révisions de documents provenant du journal.

   Dans le mode`STANDARD` autorisations d'un registre, vous devez créer des politiques IAM qui accordent des autorisations pour exécuter des instructions partiQL sur cette ressource de table. Avec des autorisations sur une ressource de table, vous pouvez exécuter des instructions qui

accèdent à l'état actuel de la table. Vous pouvez également consulter l'historique des révisions de la table à l'aide de la`history()` fonction intégrée.

Format ARN de table :

```
arn:aws:qldb:${region}:${account-id}:ledger/${ledger-name}/table/${table-id}
```

Pour de plus amples informations sur l'octroi d'autorisations sur un registre et ses ressources associées, veuillez consulterComment Amazon QLDB fonctionne avec IAM.

3. Documents

   Les tableaux se composent de révisions deDocuments QLDB, qui sont des ensembles de données`struct` au format Amazon Ion. Une révision de document représente une version unique d'une séquence de documents identifiés par un identifiant de document unique.

   QLDB stocke l'historique complet des modifications de vos documents validés. Une table vous permet de consulter l'état actuel de ses documents, tandis que la`history()` fonction vous permet de consulter l'historique complet des révisions des documents d'une table. Pour plus d'informations sur les requêtes et la rédaction de révisions, consultezUtilisation des données et de l'historique.

4. Catalogue de systèmes

   Chaque registre fournit également une ressource de catalogue définie par le système que vous pouvez interroger pour répertorier toutes les tables et tous les index d'un registre. En mode`STANDARD` autorisations d'un registre, vous devez disposer de l'`qldb:PartiQLSelect`autorisation sur cette ressource de catalogue pour effectuer les opérations suivantes :

   - ExécutezSELECT des instructions dans la table du catalogue système information_schema.user_tables.
   - Consultez les informations relatives à la table et à l'index sur la page de détails du registre de la console QLDB.
   - Consultez la liste des tables et des index dans l'éditeur PartiQL sur la console QLDB.

   Format ARN du catalogue :

   ```
   arn:aws:qldb:${region}:${account-id}:ledger/${ledger-name}/information_schema/
   user_tables
   ```

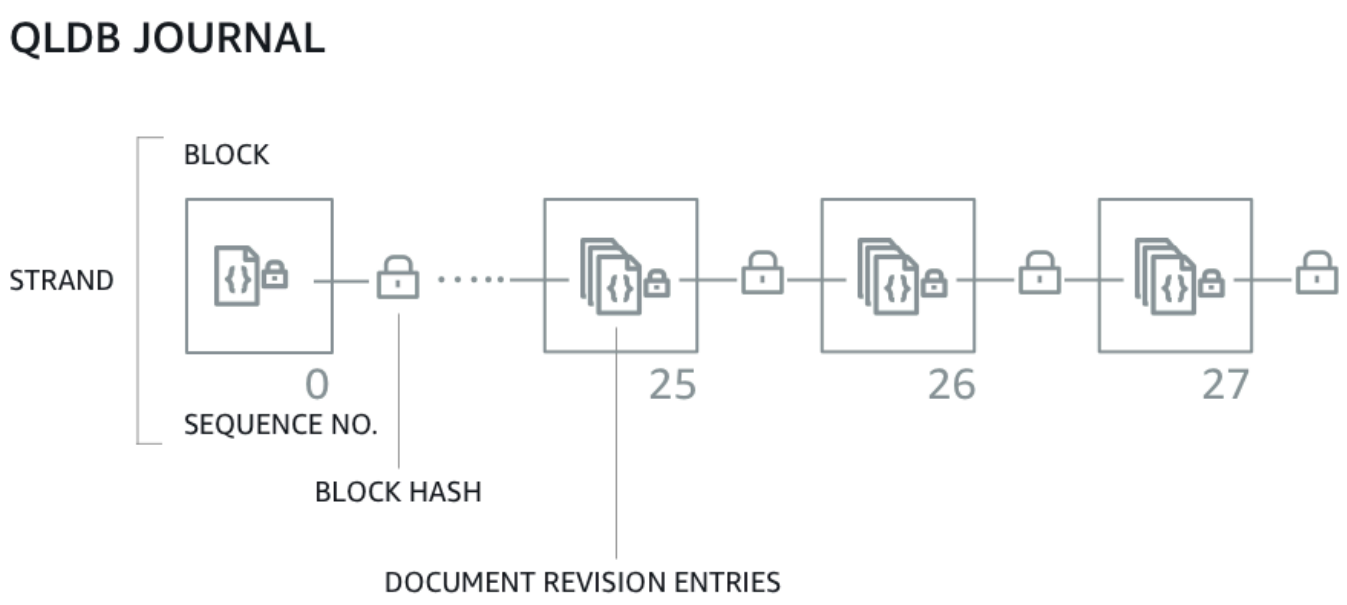
Transactions destinées à un journal

Lorsqu'une application lit ou écrit des données dans un registre QLDB, elle le fait dans une transaction de base de données. Toutes les transactions sont soumises à des limites telles que définies dansQuotas et limites d'Amazon QLDB. Dans le cadre d'une transaction, QLDB effectue les étapes suivantes :

1. Exécution de l'état actuel des données dans le registre.
2. Exécutez les instructions fournies dans la transaction, puis recherchez tout conflit à l'aide d'un contrôle de concurrence optimiste (OCC) afin de garantir une isolation entièrement sérialisable.
3. Si aucun conflit OCC n'est détecté, renvoyez les résultats de la transaction comme suit :
   - Pour les lectures, renvoyez le jeu de résultats et`SELECT` validez les instructions dans le journal uniquement en annexe.
   - Pour les écritures, validez les mises à jour, les suppressions ou les données nouvellement insérées dans le journal en les ajoutant uniquement.

Le journal représente un historique complet et immuable de toutes les modifications apportées à vos données. QLDB écrit un bloc enchaîné dans le journal lors d'une transaction. Chaque bloc contient des objets d'entrée qui représentent les révisions de documents que vous insérez, mettez à jour et supprimez, ainsi que les instructions PartiQL qui les ont validées.

Le diagramme suivant illustre cette structure de journal.



Le diagramme montre que les transactions sont enregistrées dans le journal sous forme de blocs contenant des entrées de révision du document. Chaque bloc est haché et enchaîné aux blocs suivants à des fins de vérification. Chaque bloc possède un numéro de séquence pour spécifier son adresse dans le brin.

> **Note**
>
> Dans Amazon QLDB, un brin est une partition du journal de votre registre. QLDB prend actuellement en charge les revues à un seul volet.

Pour plus d'informations sur le contenu des données d'un bloc, consultezContenu du journal dans Amazon QLDB.

Exécution de vos données

QLDB est destiné à répondre aux besoins de charge de travail de traitement transactionnel en ligne (OLTP). Un registre fournit des tableaux interrogeables de vos données en fonction des informations de transaction enregistrées dans le journal. Une vue tabulaire dans QLDB est un sous-ensemble des données d'une table. Les vues sont gérées en temps réel, de sorte qu'elles sont toujours disponibles pour que les applications puissent les interroger.

Vous pouvez interroger les vues définies par le système suivantes à l'aide d'`SELECT`instructions PartiQL :

- Utilisateur : dernière révision active uniquement des données que vous avez écrites dans le tableau (c'est-à-dire l'état actuel de vos données utilisateur). Il s'agit de la vue par défaut dans QLDB.
- Validé : dernière révision active de vos données utilisateur et des métadonnées générées par le système. Il s'agit de la table complète définie par le système qui correspond directement à votre table utilisateur.

Outre ces vues interrogeables, vous pouvez interroger l'historique des révisions de vos données à l'aide de la fonction intégréeFonction historique. La fonction d'historique renvoie à la fois vos données utilisateur et les métadonnées associées dans le même schéma que la vue validée.

Stockage de données

Il existe deux types de stockage de données dans QLDB :

- Stockage du journal : espace disque utilisé par le journal d'un grand livre. Le journal est uniquement disponible en annexe et contient l'historique complet, immuable et vérifiable de toutes les modifications apportées à vos données.
- Stockage indexé : espace disque utilisé par les tables, les index et l'historique indexé d'un registre. Le stockage indexées est constitué de données de registre ayant été optimisé pour les requêtes de transaction de transaction.

Une fois vos données validées dans le journal, elles sont matérialisées dans les tables que vous avez définies. Ces tables sont optimisées pour des requêtes plus rapides et plus efficaces. Lorsqu'une application utilise l'API de données transactionnelles pour lire des données, elle accède aux tables et aux index stockés dans votre stockage indexé.

Modèle d'API QLDB

QLDB fournit deux types d'API avec lesquelles le code de votre application peut interagir :

- Amazon QLDB : API de gestion des ressources QLDB (également appelée plan de contrôle). Cette API est utilisée uniquement pour gérer les ressources du registre et pour les opérations de données non transactionnelles. Vous pouvez utiliser ces opérations pour créer, supprimer,

répertorier et mettre à jour des registres. Vous pouvez également vérifier les données de manière cryptographique et exporter ou diffuser des blocs de journal.

- Session Amazon QLDB : API de données transactionnelles QLDB. Vous pouvez utiliser cette API pour exécuter des transactions de données sur un registre à l'aide d'instructions partiQL.

> **Important**
>
> Au lieu d'interagir directement avec l'API de session QLDB, nous vous recommandons d'utiliser le pilote QLDB ou le shell QLDB pour exécuter des transactions de données sur un registre.
>
> - Si vous travaillez avec unAWS SDK, utilisez le pilote QLDB. Le pilote fournit une couche d'abstraction de haut niveau au-dessus de l'API de données de session QLDB et gère l'`SendCommand`opération à votre place. Pour plus d'informations et une liste des langages de programmation pris en charge, consultezDémarrage.
> - Si vous utilisez leAWS CLI, utilisez le shell QLDB. Le shell est une interface de ligne de commande qui utilise le pilote QLDB pour interagir avec un registre. Pour plus d'informations, consultez Utilisation du shell Amazon QLDB (API de données uniquement).

Pour de plus amples informations sur ces opérations d'API, veuillez consulterRéférence d'API Amazon QLDB.

Étapes suivantes

Pour savoir comment utiliser un registre avec vos données, consultezUtilisation des données et de l'historique dans Amazon QLDB et suivez les exemples qui décrivent le processus de création de tables, d'insertion de données et d'exécution de requêtes de base. Ce guide explique comment ces concepts fonctionnent en profondeur, à l'aide d'exemples de données et d'exemples de requêtes pour le contexte.

Pour démarrer rapidement avec un exemple de didacticiel d'application à l'aide de la console QLDB, consultezMise en route avec la console Amazon QLDB.

Pour obtenir la liste des principaux termes et définitions décrits dans cette section, consultez leGlossaire Amazon QLDB.

Contenu du journal dans Amazon QLDB

Dans Amazon QLDB, le journal est le journal transactionnel immuable qui stocke l'historique complet et vérifiable de toutes les modifications apportées à vos données. Le journal ne peut être ajouté qu'à des fins d'ajout et se compose d'un ensemble de blocs séquencés et chaînés contenant vos données validées et d'autres métadonnées système. QLDB écrit un bloc enchaîné dans le journal lors d'une transaction.

Cette section fournit un exemple de bloc de journal contenant des exemples de données et décrit le contenu d'un bloc.

Rubriques

Exemple de bloc

Un bloc de journal contient des métadonnées de transaction ainsi que des entrées qui représentent les révisions du document qui ont été validées dans la transaction et les déclarations partiQL qui les ont validées.

Voici un exemple de bloc avec des données d'exemple.

> **Note**
>
> Cet exemple de bloc est fourni à titre informatif uniquement. Les hachages affichés ne sont pas de véritables valeurs de hachage calculées.

```
{
  blockAddress:{
    strandId:"4o5UuzWSW5PIoOGm5jPA6J",
    sequenceNo:25
  },
  transactionId:"3gtB8Q8dfIMA8lQ5pzHAMo",
```

```
  blockTimestamp:2022-06-08T18:46:46.512Z,
  blockHash:{{QS5lJt8vRxT30L9OGL5oU1pxFTe+UlEwakYBCrvGQ4A=}},
  entriesHash:{{buYYc5kV4rrRtJAsrIQnfnhgkzfQ8BKjI0C2vFnYQEw=}},
  previousBlockHash:{{I1UKRIWUgkM1X6042kcoZ/eN1rn0uxhDTc08zw9kZ5I=}},
  entriesHashList:[
    {{BUCXP6oYgmug2AfPZcAZup2lKolJNTbTuV5RA1VaFpo=}},
    {{cTIRkjuULzp/4KaUEsb/S7+TG8FvpFiZHT4tEJGcANc=}},
    {{3aktJSMyJ3C5StZv4WIJLu/w3D8mGtduZvP0ldKUaUM=}},
    {{GPKIJ1+o8mMZmPj/35ZQXoca2z64MVYMCwqs/g080IM=}}
  ],
  transactionInfo:{
    statements:[
      {
        statement:"INSERT INTO VehicleRegistration VALUE ?",
        startTime:2022-06-08T18:46:46.063Z,
        statementDigest:{{KY2nL6UGUPs5lXCLVXcUaBxcEIop0Jvk4MEjcFVBfwI=}}
      },
      {
        statement:"SELECT p_id FROM Person p BY p_id WHERE p.FirstName = ? and
p.LastName = ?",
        startTime:2022-06-08T18:46:46.173Z,
        statementDigest:{{QS2nfB8XBf2ozlDx0nvtsliOYDSmNHMYC3IRH4Uh690=}}
      },
      {
        statement:"UPDATE VehicleRegistration r SET r.Owners.PrimaryOwner.PersonId = ?
WHERE r.VIN = ?",
        startTime:2022-06-08T18:46:46.278Z,
        statementDigest:{{nGtIA9Qh0/dwIplOR8J5CTeqyUVtNUQgXfltDUo2Aq4=}}
      },
      {
        statement:"DELETE FROM DriversLicense l WHERE l.LicenseNumber = ?",
        startTime:2022-06-08T18:46:46.385Z,
        statementDigest:{{ka783dcEP58Q9AVQ1m9NOJd3JAmEvXLjzl0OjN1BojQ=}}
      }
    ],
    documents:{
      HwVFkn8IMRa0xjze5xcgga:{
        tableName:"VehicleRegistration",
        tableId:"HQZ6cgIMUi204Lq1tT4oaJ",
        statements:[0,2]
      },
      IiPTRxLGJZa342zHFCFT15:{
        tableName:"DriversLicense",
        tableId:"BvtXEB1JxZg0lJlBAtbtSV",
```

```
        statements:[3]
      }
    }
  },
  revisions:[
    {
      hash:{{FR1IWcWew0yw1TnRklo2YMF/qtwb7ohsu5FD8A4DSVg=}}
    },
    {
      blockAddress:{
        strandId:"4o5UuzWSW5PIoOGm5jPA6J",
        sequenceNo:25
      },
      hash:{{6TTHbcfIVdWoFC/j90BOZi0JdHzhjSXo1tW+uHd6Dj4=}},
      data:{
        VIN:"1N4AL11D75C109151",
        LicensePlateNumber:"LEWISR261LL",
        State:"WA",
        City:"Seattle",
        PendingPenaltyTicketAmount:90.25,
        ValidFromDate:2017-08-21,
        ValidToDate:2020-05-11,
        Owners:{
          PrimaryOwner:{
            PersonId:"3Ax20JIix5J2ulu2rCMvo2"
          },
          SecondaryOwners:[]
        }
      },
      metadata:{
        id:"HwVFkn8IMRa0xjze5xcgga",
        version:0,
        txTime:2022-06-08T18:46:46.492Z,
        txId:"3gtB8Q8dfIMA8lQ5pzHAMo"
      }
    },
    {
      blockAddress:{
        strandId:"4o5UuzWSW5PIoOGm5jPA6J",
        sequenceNo:25
      },
      hash:{{ZVF/f1uSqd5DIMqzI04CCHaCGFK/J0Jf5AFzSEk0l90=}},
      metadata:{
        id:"IiPTRxLGJZa342zHFCFT15",
```

```
        version:1,
        txTime:2022-06-08T18:46:46.492Z,
        txId:"3gtB8Q8dfIMA8lQ5pzHAMo"
      }
    }
  ]
}
```

Dans lerevisions champ, certains objets de révision peuvent uniquement contenir unehash valeur et aucun autre attribut. Il s'agit de révisions du système uniquement internes qui ne contiennent pas de données utilisateur. Les hachages de ces révisions font partie de la chaîne de hachage complète de la revue, qui est requise pour la vérification cryptographique.

Contenu du bloc

Un bloc de journal contient les champs suivants :

`blockAddress`

L'emplacement du bloc dans le journal. Une adresse est une structure Amazon Ion qui comporte deux champs :`strandId` et`sequenceNo`.

Par exemple : `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:14}`

`transactionId`

ID unique de la transaction à l'origine du blocage.

`blockTimestamp`

L'horodatage auquel le bloc a été enregistré dans le journal.

`blockHash`

La valeur de hachage de 256 bits qui représente de manière unique le bloc. C'est le hachage de la concaténation de`entriesHash` et`previousBlockHash`.

`entriesHash`

Le hachage qui représente toutes les entrées du bloc, y compris les entrées du système uniquement internes. Il s'agit du hachage racinaire de l'arbre Merkle dans lequel les nœuds foliaires sont composés de tous les hachages`entriesHashList`.

`previousBlockHash`

Le hachage du bloc enchaîné précédent dans le journal.

`entriesHashList`

La liste des hachages qui représentent chaque entrée du bloc. Cette liste peut inclure les hachages d'entrée suivants :

- Le hachage ionique qui représente`transactionInfo`. Cette valeur est calculée en prenant le hachage ionique de l'ensemble de la`transactionInfo` structure.
- Le hachage racinaire de l'arbre Merkle dans lequel les nœuds foliaires sont composés de tous les hachages`revisions`.
- Le hachage ionique qui représente`redactionInfo`. Ce hachage n'existe que dans les blocs qui ont été validés par une transaction de rédaction. Sa valeur est calculée en prenant le hachage ionique de l'ensemble de la`redactionInfo` structure.
- Des hachages qui représentent les métadonnées internes du système uniquement. Ces hachages peuvent ne pas exister dans tous les blocs.

`transactionInfo`

Une structure Amazon Ion qui contient des informations sur les déclarations de la transaction qui a validé le blocage. Cette structure possède les champs suivants :

- `statements`— La liste des instructions PartiQL et leur`startTime` date de début d'exécution. Chaque instruction possède un`statementDigest` hachage, qui est nécessaire pour calculer le hachage de la`transactionInfo` structure.
- `documents`— Les identifiants de documents qui ont été mis à jour par les déclarations. Chaque document inclut le`tableName` et auquel il appartient, ainsi`tableId` que l'index de chaque déclaration qui l'a mis à jour.

`revisions`

La liste des révisions de documents qui ont été validées dans le bloc. Chaque structure de révision contient tous les champs de la vue validée de la révision.

Cela peut également inclure des hachages qui représentent des révisions internes uniquement du système qui font partie de la chaîne de hachage complète d'un journal.

Révisions expurgées

Dans Amazon QLDB, une`DELETE` instruction ne supprime logiquement un document qu'en créant une nouvelle révision qui le marque comme supprimé. QLDB prend également en charge une

opération de suppression des données qui vous permet de supprimer définitivement les révisions de documents inactives dans l'historique d'un tableau.

L'opération de rédaction supprime uniquement les données utilisateur dans la révision spécifiée et laisse la séquence du journal et les métadonnées du document inchangées. Cela préserve l'intégrité globale des données de votre registre. Pour plus d'informations et pour voir un exemple d'opération de rédaction, consultezRédaction de révisions de documents.

Exemple de révision expurgée

Prenons l'exemple de bloc précédent. Dans ce bloc, supposons que vous supprimiez la révision dont l'ID de document`HwVFkn8IMRa0xjze5xcgga` et le numéro de version sont`0`.

Une fois la rédaction terminée, les données utilisateur de la révision (représentées par la`data` structure) sont remplacées par un nouveau`dataHash` champ. La valeur de ce champ est le hachage ionique de la`data` structure supprimée. Par conséquent, le registre conserve l'intégrité globale de ses données et reste vérifiable cryptographiquement par le biais des opérations d'API de vérification existantes.

L'exemple de révision suivant montre les résultats de cette rédaction, le nouveau`dataHash` champ étant *`surligné en italiques rouges`*.

> **Note**
>
> Cet exemple de révision est fourni à titre informatif uniquement. Les hachages affichés ne sont pas de véritables valeurs de hachage calculées.

```
...
{
  blockAddress:{
    strandId:"4o5UuzWSW5PIoOGm5jPA6J",
    sequenceNo:25
  },
  hash:{{6TTHbcfIVdWoFC/j90BOZi0JdHzhjSXo1tW+uHd6Dj4=}},
  dataHash:{{s83jd7sfhsdfhksj7hskjdfjfpIPP/DP2hvionas2d4=}},
  metadata:{
    id:"HwVFkn8IMRa0xjze5xcgga",
    version:0,
    txTime:2022-06-08T18:46:46.492Z,
```

```
    txId:"3gtB8Q8dfIMA8lQ5pzHAMo"
  }
}
...
```

QLDB ajoute également un nouveau bloc au journal pour la demande de rédaction terminée. Ce bloc contient une`redactionInfo` entrée supplémentaire contenant la liste des révisions supprimées lors de la transaction, comme le montre l'exemple suivant.

```
...
redactionInfo:{
  revisions:[
    {
      blockAddress:{
        strandId:"4o5UuzWSW5PIoOGm5jPA6J",
        sequenceNo:25
      },
      tableId:"HQZ6cgIMUi204Lq1tT4oaJ",
      documentId:"HwVFkn8IMRa0xjze5xcgga",
      version:0
    }
  ]
}
...
```

Exemple d'application

Pour un exemple de code Java qui valide la chaîne de hachage d'un journal à l'aide de données exportées, consultez le GitHub référentiel aws-samples/amazon-qldb-dmv-sample -java. Cet exemple d'application inclut les fichiers de classe suivants :

- ValidateQldbHashChain.java — Contient un code de didacticiel qui exporte des blocs de journal depuis un registre et utilise les données exportées pour valider la chaîne de hachage entre les blocs.
- JournalBlock.java — Contient une méthode nommée`verifyBlockHash()` qui montre comment calculer chaque composant de hachage individuel dans un bloc. Cette méthode est appelée par le code du didacticiel dans`ValidateQldbHashChain.java`.

Pour obtenir des instructions sur le téléchargement et l'installation de cet exemple d'application complet, consultezInstallation de l'exemple d'application Java Amazon QLDB. Avant d'exécuter

le code du didacticiel, assurez-vous de suivre les étapes 1 à 3 duTutoriel Java pour configurer un registre d'échantillons et le charger avec des exemples de données.

Consulter aussi

Pour plus d'informations sur les revues dans QLDB, consultez les rubriques suivantes :

- Exportation de données de journal depuis Amazon QLDB— Pour savoir comment exporter les données d'un journal vers Amazon Simple Storage Service (Amazon S3).
- Diffusion en continu de données de journaux depuis Amazon QLDB— Pour savoir comment diffuser les données d'un journal sur Amazon Kinesis Data Streams.
- Vérification des données dans Amazon QLDB— Pour en savoir plus sur la vérification cryptographique des données de journaux.

Glossaire Amazon QLDB

Vous trouverez ci-dessous la définition de termes clés que vous pourriez rencontrer dans le cadre de l'utilisation d'Amazon QLDB.

bloc | digérer | document | ID du document | révision du document | entrée | field | index | stockage indexé | journal | bloc de journal | stockage de journaux | volet du journal | conseil pour le journal | grand livre | preuve | révision | séance | brin | table | vue du tableau | voir

bloc

Objet enregistré dans le journal dans le cadre d'une transaction. Une seule transaction écrit un bloc dans le journal, de sorte qu'un bloc ne peut être associé qu'à une seule transaction. Un bloc contient des entrées qui représentent les révisions du document qui ont été validées lors de la transaction, ainsi que les instructions PartiQL qui les ont validées.

Chaque bloc possède également une valeur de hachage à des fins de vérification. Un hachage de bloc est calculé à partir des hachages d'entrée de ce bloc combinés au hachage du bloc chaîné précédent.

digérer

Une valeur de hachage de 256 bits qui représente de manière unique l'historique complet des révisions de documents de votre registre à un moment donné. Un condensé est calculé à partir de la chaîne de hachage complète de votre journal en fonction du dernier bloc validé dans le journal à ce moment-là.

QLDB vous permet de générer un condensé sous forme de fichier de sortie sécurisé. Vous pouvez ensuite utiliser ce fichier de sortie pour vérifier l'intégrité des révisions de vos documents par rapport à ce hachage.

document

Un ensemble de données `austruct` format Amazon Ion qui peut être inséré, mis à jour et supprimé dans un tableau. Un document QLDB peut contenir des données structurées, semi-structurées, imbriquées et sans schéma.

ID du document

Identifiant universel unique (UUID) que QLDB attribue à chaque document que vous insérez dans un tableau. Cet identifiant est un nombre de 128 bits qui est représenté dans une chaîne alphanumérique codée en Base62 d'une longueur fixe de 22 caractères.

révision du document

Structure ionique qui représente une version unique d'une séquence de documents identifiés par un identifiant de document unique. Une révision inclut à la fois vos données utilisateur (c'est-à-dire les données que vous avez écrites dans le tableau) et les métadonnées générées par le système. Chaque révision est associée à un tableau et est identifiée de manière unique par une combinaison de l'identifiant du document et d'un numéro de version en base zéro.

entrée

Objet contenu dans un bloc. Les entrées représentent les révisions de documents qui sont insérées, mises à jour et supprimées dans une transaction, ainsi que les instructions PartiQL qui les ont validées.

Chaque entrée possède également une valeur de hachage à des fins de vérification. Un hachage d'entrée est calculé à partir des hachages de révision ou des hachages d'instructions contenus dans cette entrée.

field

Une paire nom-valeur qui constitue chaque attribut d'un document QLDB. Le nom est un symbole et sa valeur n'est pas restreinte.

index

Structure de données que vous pouvez créer sur une table pour optimiser les performances des opérations de récupération de données. Pour plus d'informations sur les index dans QLDB, consultezCREATE INDEX la référence Amazon QLDB PartiQL.

stockage indexé

L'espace disque utilisé par les tables, les index et l'historique indexé d'un registre. Le stockage indexé se compose de données de registre optimisées pour requêtes hautes performances.

journal

L'ensemble enchaîné de tous les blocs qui sont validés dans votre registre. Le journal est uniquement disponible en annexe et représente un historique complet et immuable de toutes les modifications apportées aux données de votre registre.

bloc de journal

Consultez bloc.

stockage de journaux

L'espace disque utilisé par le journal d'un grand livre.

volet du journal

Consultez brin.

conseil pour le journal

Bloc le plus récent dans un journal à un moment spécifique.

grand livre

Instance d'une ressource de base de données Amazon QLDB Ledger. Il s'agit du type deAWS ressource principal dans QLDB. Un grand livre comprend à la fois un stockage de journaux et un stockage indexé. Une fois que les données du registre sont enregistrées dans le journal, elles peuvent être consultées dans les tableaux des révisions des documents Amazon Ion.

preuve

La liste ordonnée de valeurs de hachage de 256 bits que QLDB renvoie pour un condensé et une révision de document donnés. Il comprend les hachages requis par un modèle d'arbre Merkle pour enchaîner le hachage de révision donné au hachage de synthèse. Vous utilisez une preuve pour vérifier l'intégrité de vos révisions par rapport au résumé. Pour plus d'informations, veuillez consulter Vérification des données dans Amazon QLDB.

révision

Consultez révision du document.

séance

Objet qui gère les informations relatives à vos demandes de transactions de données et à vos réponses à destination et en provenance d'un registre. Une session active (une session qui exécute activement une transaction) représente une connexion unique à un registre. QLDB prend en charge une transaction active par session.

brin

Partition d'un journal. QLDB prend actuellement en charge les revues à un seul volet.

table

Vue matérialisée d'un ensemble non ordonné de révisions de documents validées dans le journal du grand livre.

vue du tableau

Sous-ensemble interrogeable des données d'un tableau, basé sur les transactions validées dans le journal. Dans une instruction PartiQL, une vue est désignée par un qualificatif de préfixe (commençant par`_ql_`) pour le nom d'une table.

Vous pouvez interroger les vues définies par le système suivantes à l'aide d'`SELECT`instructions :

- Utilisateur : dernière révision active uniquement des données que vous avez écrites dans le tableau (c'est-à-dire l'état actuel de vos données utilisateur). Il s'agit de la vue par défaut dans QLDB.
- Validé : dernière révision active de vos données utilisateur et des métadonnées générées par le système. Il s'agit de la table complète définie par le système qui correspond directement à votre table utilisateur. Par exemple : `_ql_committed_`*`TableName`*.

voir

Consultez vue du tableau.

Accès à Amazon QLDB

Vous pouvez accéder à Amazon QLDB à l'aide de l'API, AWS Management Console de AWS Command Line Interface la AWS CLI() ou de QLDB. Les sections suivantes décrivent comment utiliser ces options et les conditions requises pour les utiliser.

Prérequis

Avant de pouvoir accéder à QLDB, vous devez en configurer Compte AWS un si ce n'est déjà fait.

Rubriques

- Inscrivez-vous pour un Compte AWS
- Création d'un utilisateur doté d'un accès administratif
- Gérer les autorisations QLDB dans IAM
- Accorder l'accès programmatique (facultatif)

Inscrivez-vous pour un Compte AWS

Si vous n'en avez pas Compte AWS, procédez comme suit pour en créer un.

Pour vous inscrire à un Compte AWS

1. Ouvrez https://portal.aws.amazon.com/billing/signup.
2. Suivez les instructions en ligne.

   Dans le cadre de la procédure d‘inscription, vous recevrez un appel téléphonique et vous saisirez un code de vérification en utilisant le clavier numérique du téléphone.

   Lorsque vous vous inscrivez à un Compte AWS, un Utilisateur racine d'un compte AWSest créé. Par défaut, seul l‘utilisateur racine a accès à l‘ensemble des Services AWS et des ressources de ce compte. Pour des raisons de sécurité, attribuez un accès administratif à un utilisateur et utilisez uniquement l'utilisateur root pour effectuer les tâches nécessitant un accès utilisateur root.

AWS vous envoie un e-mail de confirmation une fois le processus d'inscription terminé. Vous pouvez afficher l'activité en cours de votre compte et gérer votre compte à tout moment en accédant à https://aws.amazon.com/ et en choisissant Mon compte.

Création d'un utilisateur doté d'un accès administratif

Après vous être inscrit à un Compte AWS, sécurisez Utilisateur racine d'un compte AWS AWS IAM Identity Center, activez et créez un utilisateur administratif afin de ne pas utiliser l'utilisateur root pour les tâches quotidiennes.

Sécurisez votre Utilisateur racine d'un compte AWS

1. Connectez-vous en AWS Management Consoletant que propriétaire du compte en choisissant Utilisateur root et en saisissant votre adresse Compte AWS e-mail. Sur la page suivante, saisissez votre mot de passe.

   Pour obtenir de l'aide pour vous connecter en utilisant l'utilisateur racine, consultez Connexion en tant qu'utilisateur racine dans le Guide de l'utilisateur Connexion à AWS .
2. Activez l'authentification multifactorielle (MFA) pour votre utilisateur racine.

   Pour obtenir des instructions, voir Activer un périphérique MFA virtuel pour votre utilisateur Compte AWS root (console) dans le guide de l'utilisateur IAM.

Création d'un utilisateur doté d'un accès administratif

1. Activez IAM Identity Center.

   Pour obtenir des instructions, consultez Activation d' AWS IAM Identity Center dans le Guide de l'utilisateur AWS IAM Identity Center .
2. Dans IAM Identity Center, accordez un accès administratif à un utilisateur.

   Pour un didacticiel sur l'utilisation du Répertoire IAM Identity Center comme source d'identité, voir Configurer l'accès utilisateur par défaut Répertoire IAM Identity Center dans le Guide de AWS IAM Identity Center l'utilisateur.

Connectez-vous en tant qu'utilisateur disposant d'un accès administratif

- Pour vous connecter avec votre utilisateur IAM Identity Center, utilisez l'URL de connexion qui a été envoyée à votre adresse e-mail lorsque vous avez créé l'utilisateur IAM Identity Center.

Pour obtenir de l'aide pour vous connecter en utilisant un utilisateur d'IAM Identity Center, consultez la section Connexion au portail AWS d'accès dans le guide de l'Connexion à AWS utilisateur.

Attribuer l'accès à des utilisateurs supplémentaires

1. Dans IAM Identity Center, créez un ensemble d'autorisations conforme aux meilleures pratiques en matière d'application des autorisations du moindre privilège.

   Pour obtenir des instructions, voir Création d'un ensemble d'autorisations dans le guide de AWS IAM Identity Center l'utilisateur.
2. Affectez des utilisateurs à un groupe, puis attribuez un accès d'authentification unique au groupe.

   Pour obtenir des instructions, voir Ajouter des groupes dans le guide de AWS IAM Identity Center l'utilisateur.

Gérer les autorisations QLDB dans IAM

Pour plus d'informations sur l'utilisation de AWS Identity and Access Management (IAM) pour gérer les autorisations QLDB pour les utilisateurs, consultez. Comment Amazon QLDB fonctionne avec IAM

Accorder l'accès programmatique (facultatif)

Les utilisateurs ont besoin d'un accès programmatique s'ils souhaitent interagir avec AWS l'extérieur du AWS Management Console. La manière d'accorder un accès programmatique dépend du type d'utilisateur qui y accède AWS.

Pour accorder aux utilisateurs un accès programmatique, choisissez l’une des options suivantes.

Quel utilisateur a besoin d’un accès programmatique ?	Pour	Par
Identité de la main-d’œuvre	Utilisez des informations d'identification temporaires pour signer les demandes	Suivez les instructions de l’interface que vous souhaitez utiliser.

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Par
(Utilisateurs gérés dans IAM Identity Center)	programmatiques adressées aux AWS CLI AWS SDK ou AWS aux API.	• Pour le AWS CLI, voir Configuration du AWS CLI à utiliser AWS IAM Identity Center dans le guide de AWS Command Line Interface l'utilisateur. • Pour les AWS SDK, les outils et les AWS API, consultez la section Authentification IAM Identity Center dans le Guide de référence AWS des SDK et des outils.
IAM	Utilisez des informations d'identification temporaires pour signer les demandes programmatiques adressées aux AWS CLI AWS SDK ou AWS aux API.	Suivez les instructions de la section Utilisation d'informations d'identification temporaires avec AWS les ressources du Guide de l'utilisateur IAM.

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Par
IAM	(Non recommandé) Utilisez des informations d'identification à long terme pour signer les AWS CLI demandes programmatiques adressées aux AWS SDK ou AWS aux API.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour le AWS CLI, voir Authentification à l'aide des informations d'identification utilisateur IAM dans le Guide de l'AWS Command Line Interface utilisateur. • Pour les AWS SDK et les outils, voir Authentifier à l'aide d'informations d'identification à long terme dans le Guide de AWS référence des SDK et des outils. • Pour les AWS API, consultez la section Gestion des clés d'accès pour les utilisateurs IAM dans le guide de l'utilisateur IAM.

Comment accéder à Amazon QLDB

Après avoir rempli les conditions requises pour configurer un Compte AWS, consultez les rubriques suivantes pour en savoir plus sur l'accès à QLDB :

- Utilisation de la console
- Utilisation de AWS CLI (API de gestion uniquement)
- Utilisation du shell Amazon QLDB (API de données uniquement)
- Utilisation de l'API

Accès à Amazon QLDB à l'aide de la console

Vous pouvez accéder au AWS Management Console pour Amazon QLDB à l'adresse https://console.aws.amazon.com/qldb.

Vous pouvez utiliser la console pour effectuer les opérations suivantes dans QLDB :

- Créez, supprimez, décrivez et listez des registres.
- Exécutez les instructions partiQL à l'aide de l'éditeur partiQL.
- Gérez les balises pour les ressources QLDB.
- Vérifiez les données du journal de manière cryptographique.
- Exportez ou diffusez des blocs de journal.

Pour savoir comment créer un registre Amazon QLDB et le configurer avec des exemples de données d'application, consultez. Mise en route avec la console Amazon QLDB

Référence rapide de l'éditeur PartiQL

Amazon QLDB prend en charge un sous-ensemble de partiQL comme langage de requête et Amazon Ion comme format de données orienté document. Pour un guide complet et des informations plus détaillées sur l'implémentation QLDB de partiQL, consultez le. Référence Amazon QLDB

Les rubriques suivantes fournissent un bref aperçu de la manière d'utiliser partiQL dans QLDB.

Rubriques

- Conseils rapides sur PartiQL dans QLDB
- Commandes
- Vues définies par le système
- Règles de syntaxe de base
- Raccourcis clavier de l'éditeur PartiQL

Conseils rapides sur PartiQL dans QLDB

Voici un bref résumé des conseils et des meilleures pratiques pour travailler avec partiQL dans QLDB :

- Comprenez la simultanéité et les limites de transaction — Toutes les déclarations, y compris les `SELECT` requêtes, sont soumises à des conflits optimistes en matière de contrôle simultané (OCC) et à des limites de transaction, y compris un délai d'expiration de 30 secondes.
- Utiliser des index : utilisez des index à cardinalité élevée et exécutez des requêtes ciblées pour optimiser vos instructions et éviter d'analyser des tables complètes. Pour en savoir plus, veuillez consulter la section Optimisation des performances des données.
- Utiliser des prédicats d'égalité : les recherches indexées nécessitent un opérateur d'égalité (ou)=. `IN` Les opérateurs d'inégalité (`<>`,`LIKE`,,`BETWEEN`) ne sont pas éligibles aux recherches indexées et entraînent une analyse complète des tables.
- Utilisez uniquement les jointures internes : QLDB ne prend en charge que les jointures internes. Il est recommandé de joindre des champs indexés pour chaque table que vous joignez. Choisissez des indices de cardinalité élevés pour les critères de jointure et les prédicats d'égalité.

Commandes

QLDB prend en charge les commandes partiQL suivantes.

Langage de définition de données (DDL)

Command	Description
CREATE INDEX	Crée un index pour un champ de document de niveau supérieur sur une table.
CREATE TABLE	Crée une table.
DROP INDEX	Supprime un index d'une table.
DROP TABLE	Désactive une table existante.
TABLEAU DE DÉBALLAGE	Réactive une table inactive.

Langage de manipulation de données (DML)

Command	Description
DELETE	Marque un document actif comme supprimé en créant une nouvelle révision finale du document.
DE (INSÉRER, SUPPRIMER ou DÉFINIR)	Sémantiquement identique à. UPDATE
INSERT	Ajoute un ou plusieurs documents à un tableau.
SELECT	Récupère les données d'une ou de plusieurs tables.
MISE A JOUR	Met à jour, insère ou supprime des éléments spécifiques dans un document.

Exemples de déclarations DML

INSÉRER

```
INSERT INTO VehicleRegistration VALUE
{
    'VIN' : 'KM8SRDHF6EU074761', --string
    'RegNum' : 1722, --integer
    'PendingPenaltyTicketAmount' : 130.75, --decimal
    'Owners' : { --nested struct
        'PrimaryOwner' : { 'PersonId': '294jJ3YUoH1IEEm8GSabOs' },
        'SecondaryOwners' : [ --list of structs
            { 'PersonId' : '1nmeDdLo3AhGswBtyM1eYh' },
            { 'PersonId': 'IN7MvYtUjkp1GMZu0F6CG9' }
        ]
    },
    'ValidToDate' : `2020-06-25T` --Ion timestamp literal with day precision
}
```

METTRE À JOUR ET INSÉRER

```
UPDATE Vehicle AS v
INSERT INTO v VALUE 26500 AT 'Mileage'
```

```
WHERE v.VIN = '1N4AL11D75C109151'
```

METTRE À JOUR/SUPPRIMER

```
UPDATE Person AS p
REMOVE p.Address
WHERE p.GovId = '111-22-3333'
```

SELECT — Sous-requête corrélée

```
SELECT r.VIN, o.SecondaryOwners
FROM VehicleRegistration AS r, @r.Owners AS o
WHERE r.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

SELECT — Jointure interne

```
SELECT v.Make, v.Model, r.Owners
FROM VehicleRegistration AS r INNER JOIN Vehicle AS v
ON r.VIN = v.VIN
WHERE r.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

SELECT — Obtenir l'identifiant du document à l'aide de la clause BY

```
SELECT r_id FROM VehicleRegistration AS r BY r_id
WHERE r.VIN = '1HVBBAANXWH544237'
```

Vues définies par le système

QLDB prend en charge les vues définies par le système suivantes d'une table.

Vue	Description
table_name	Vue utilisateur par défaut d'un tableau qui inclut uniquement l'état actuel de vos données utilisateur.
_ql_committed_ *table_name*	Vue validée complète définie par le système d'une table qui inclut l'état actuel de vos données utilisateur et des métadonnées générées par le système, telles qu'un identifiant de document.

Vue	Description
`history(table_name )`	Fonction d'historique intégrée qui renvoie l'historique complet des révisions d'une table.

Règles de syntaxe de base

QLDB prend en charge les règles de syntaxe de base suivantes pour partiQL.

Caractère	Description
'	Les guillemets simples indiquent des valeurs de chaîne ou des noms de champs dans les structures Amazon Ion.
"	Les guillemets doubles indiquent des identifiants entre guillemets, tels qu'un mot réservé utilisé comme nom de table.
`	Les backticks indiquent les valeurs littérales des ions.
.	La notation par points permet d'accéder aux noms de champs d'une structure parent.
[ ]	Les crochets définissent un ion `list` ou indiquent un nombre ordinal basé sur zéro pour une liste existante.
{ }	Les bretelles bouclées définissent un ion. `struct`
<< >>	Les crochets à double angle définissent un sac partiQL, qui est une collection non ordonnée. Vous utilisez un sac pour insérer plusieurs documents dans un tableau.
Sensibilité à la casse	Tous les noms d'objets du système QLDB, y compris les noms de champs et de tables, distinguent les majuscules et minuscules.

Raccourcis clavier de l'éditeur PartiQL

L'éditeur PartiQL de la console QLDB prend en charge les raccourcis clavier suivants.

Action	macOS	Windows
Exécuter	Cmd+Return	Ctrl+Enter
Comment	Cmd+/	Ctrl+/
Effacer	Cmd+Shift+Delete	Ctrl+Shift+Delete

Accès à Amazon QLDB à l'aide de (API AWS CLI de gestion uniquement)

Vous pouvez utiliser le AWS Command Line Interface (AWS CLI) pour contrôler plusieurs Services AWS depuis la ligne de commande et les automatiser par le biais de scripts. Vous pouvez l'utiliser AWS CLI pour des opérations ponctuelles selon vos besoins. Vous pouvez également l'utiliser pour intégrer des opérations Amazon QLDB dans des scripts utilitaires.

Pour accéder à la CLI, vous avez besoin d'un ID de clé d'accès et d'une clé d'accès secrète. Utilisation des informations d'identification temporaires au lieu des clés d'accès à long terme si possible. Les informations d'identification temporaires incluent un ID de clé d'accès, une clé d'accès secrète et un jeton de sécurité qui indique la date d'expiration des informations d'identification. Pour plus d'informations, consultez la section Utilisation d'informations d'identification temporaires avec AWS des ressources dans le Guide de l'utilisateur IAM.

Pour une liste complète et des exemples d'utilisation de toutes les commandes disponibles pour QLDB dans AWS CLI le, consultez AWS CLI la référence des commandes.

> **Note**
>
> Le AWS CLI seul prend en charge les opérations d'API de `qldb` gestion répertoriées dans leRéférence d'API Amazon QLDB. Cette API est utilisée uniquement pour gérer les ressources du registre et pour les opérations de données non transactionnelles.
> Pour exécuter des transactions de données avec l'`qldb-session`API à l'aide d'une interface de ligne de commande, consultezAccès à Amazon QLDB à l'aide du shell QLDB (API de données uniquement).

Rubriques

- Installation et configuration du AWS CLI
- Utilisation du AWS CLI avec QLDB

Installation et configuration du AWS CLI

Il AWS CLI fonctionne sous Linux, macOS ou Windows. Pour l'installer et le configurer, consultez les instructions suivantes du guide de l'AWS Command Line Interface utilisateur :

1. Installation ou mise à jour de la dernière version du AWS CLI
2. Configuration rapide

Utilisation du AWS CLI avec QLDB

Le format de ligne de commande consiste en un nom d'opération Amazon QLDB, suivi des paramètres de cette opération. Il AWS CLI prend en charge une syntaxe abrégée pour les valeurs des paramètres, en plus du JSON.

Utilisez `help` pour répertorier toutes les commandes disponibles dans QLDB :

```
aws qldb help
```

Vous pouvez également utiliser `help` pour décrire une commande spécifique et en savoir plus sur son utilisation :

```
aws qldb create-ledger help
```

Par exemple, pour créer un registre :

```
aws qldb create-ledger --name my-example-ledger --permissions-mode STANDARD
```

Accès à Amazon QLDB à l'aide du shell QLDB (API de données uniquement)

Amazon QLDB fournit un shell de ligne de commande pour interagir avec l'API de données transactionnelles. Avec le shell QLDB, vous pouvez exécuter des instructions PartiQL sur des données de registre.

La dernière version de ce shell est écrite en Rust et est open source dans le GitHub référentiel awslabs/amazon-qldb-shell sur la`main` branche par défaut. La version Python (v1) est également toujours disponible pour une utilisation dans le même référentiel de la`master` branche.

> **Note**
>
> Le shell Amazon QLDB prend uniquement en charge l'API de données`qldb-session` transactionnelles. Cette API est uniquement utilisée pour exécuter des instructions partiQL sur un registre QLDB.
> Pour interagir avec les opérations de l'API`qldb` de gestion à l'aide d'une interface de ligne de commande, consultezAccès à Amazon QLDB à l'aide de (API AWS CLI de gestion uniquement).

Cet outil n'est pas destiné à être intégré à une application ou adopté à des fins de production. L'objectif de cet outil est de vous permettre d'expérimenter rapidement QLDB et PartiQL.

Les sections suivantes vous décrivent comment commencer à utiliser les shell QLDB.

Rubriques

Prérequis

Avant de commencer à utiliser QLDB, vous devez exécuter les opérations shell QLDB, procédez comme suit :

1. Suivez les instructions deAWS configuration dansAccès à Amazon QLDB. Cela inclut les éléments suivants :

   1. S'inscrire àAWS.
   2. Créer un utilisateur doté des autorisations QLDB appropriées.
   3. Accorder un accès par programmation à des fins de développement.

2. Configurez vosAWS informations d'identification et celles par défautRégion AWS. Pour obtenir des instructions, reportez-vous à la section Principes de base de la configuration dans le Guide deAWS Command Line Interface l'utilisateur.

   Pour obtenir la liste complète des régions disponibles, consultez les points de terminaison et les quotas Amazon QLDB dans le Références générales AWS.

3. Pour tous les registres en mode`STANDARD` autorisations, créez des politiques IAM qui vous autorisent à exécuter des instructions partiQL sur les tables appropriées. Pour découvrir comment créer ces politiques, veuillez consulterCommencer à utiliser le mode d'autorisation standard dans Amazon QLDB.

Installation de la coque

Pour installer la dernière version du shell QLDB, consultez le fichier README.md sur GitHub. QLDB fournit des fichiers binaires prédéfinis pour Linux, macOS et Windows dans la section Releases du GitHub référentiel.

Pour macOS, le shell s'intègre au`aws/tap` Homebrew Tap. Pour installer le shell macOS à l'aide de Homebrew, exécutez les commandes suivantes.

```
$ xcode-select --install # Required to use Homebrew
$ brew tap aws/tap # Add AWS as a Homebrew tap
$ brew install qldbshell
```

Configuration

Après l'installation, le shell charge le fichier de configuration par défaut qui se trouve`$XDG_CONFIG_HOME/qldbshell/config.ion` lors de l'initialisation. Sous Linux et macOS, ce fichier se trouve généralement à l'adresse`~/.config/qldbshell/config.ion`. Si un tel fichier n'existe pas, le shell s'exécute avec les paramètres par défaut.

Vous pouvez créer un`config.ion` fichier manuellement après l'installation. Ce fichier de configuration utilise le format de données Amazon Ion. Voici un exemple de`config.ion` fichier minimal.

```
{
  default_ledger: "my-example-ledger"
}
```

S'il`default_ledger` n'est pas défini dans votre fichier de configuration, le`--ledger` paramètre est obligatoire lorsque vous appelez le shell. Pour obtenir la liste complète des options de configuration, consultez le fichier README.md sur GitHub.

Invoquer la coque

Pour appeler le shell QLDB sur votre terminal de ligne de commande pour un registre spécifique, exécutez la commande suivante. Remplacez *`my-example-ledger`*par le nom de votre registre.

```
$ qldb --ledger my-example-ledger
```

Cette commande se connecte à votre valeur par défautRégion AWS. Pour spécifier explicitement la région, vous pouvez exécuter la commande avec le`--qldb-session-endpoint` paramètre`--region` or, comme décrit dans la section suivante.

Après avoir appelé une session`qldb` shell, vous pouvez saisir les types de saisie suivants :

- Commandes Shell
- Instructions partiQL uniques dans des transactions distinctes
- Plusieurs instructions partiQL au sein d'une transaction

Paramètres de la coque

Pour obtenir la liste complète des indicateurs et des options disponibles pour appeler un shell, exécutez la`qldb` commande avec l'`--help`indicateur, comme suit.

```
$ qldb --help
```

Vous trouverez ci-dessous quelques indicateurs clés et options pour la`qldb` commande. Vous pouvez ajouter ces paramètres facultatifs pour remplacer le profil d'informations d'identificationRégion AWS, le point de terminaison, le format des résultats et d'autres options de configuration.

Utilisation

```
$ qldb [FLAGS] [OPTIONS]
```

DRAPEAUX

`-h`, **`--help`**

Imprime les informations d'aide.

`-v`, **`--verbose`**

Configure la verbosité de la journalisation. Par défaut, le shell enregistre uniquement les erreurs. Pour augmenter le niveau de verbosité, répétez cet argument (par exemple,`-vv`). Le niveau le plus élevé`-vvv` correspond à la`trace` verbosité.

`-V`, **`--version`**

Imprime les informations de version.

OPTIONS

`-l`,**`--ledger`** ***`NOM DU REGISTRE`***

Le nom du registre auquel se connecter. Il s'agit d'un paramètre de shell obligatoire s'il`default_ledger` n'est pas défini dans votre`config.ion` fichier. Dans ce fichier, vous pouvez définir des options supplémentaires, telles que la région.

`-c`,**`--config`** ***`FICHIER DE CONFIGURATION`***

Le fichier dans lequel vous pouvez définir toutes les options de configuration du shell. Pour plus de détails sur le formatage et une liste complète des options de configuration, consultez le fichier README.md sur GitHub.

`-f`, **`--format ion|table`**

Format de sortie des résultats de votre requête. La valeur par défaut est `ion`.

`-p`,**`--profile`** ***`PROFIL`***

L'emplacement de votre profilAWS d'informations d'identification à utiliser pour l'authentification.

S'il n'est pas fourni, le shell utilise votreAWS profil par défaut, qui se trouve à l'adresse`~/.aws/credentials`.

`-r`,`--region` ***`REGION_CODE`***

Région AWSCode du registre QLDB auquel se connecter. Par exemple : `us-east-1`.

S'il n'est pas fourni, le shell se connecte à votre configuration par défaut,Région AWS comme indiqué dans votreAWS profil.

`-s`,`--qldb-session-endpoint` ***`QLDB_SESSION_ENDPOINT`***

Le point de terminaison de l'`qldb-session`API auquel se connecter.

Pour obtenir la liste complète des régions et points de terminaison QLDB et quotas Amazon QLDB Amazon QLDB Amazon QLDB Amazon QLDB Amazon QLDB Amazon QLDB Amazon QLDB Amazon QLDB Amazon QLDB Références générales AWS

Référence de commande

Une fois que vous avez appelé une`qldb` session, le shell prend en charge les clés et les commandes de base de données suivantes :

Clés Shell

Clé	Description de la fonction
Enter	Exécute la déclaration.
Escape+Enter (macOS, Linux) Shift+Enter (Windows)	Commence une nouvelle ligne pour saisir une instruction qui s'étend sur plusieurs lignes. Vous pouvez également copier le texte d'entrée comportant plusieurs lignes et le coller dans le shell. Pour obtenir des instructions sur la configura tionOption plutôt qu'enEscape tant que clé méta dans macOS, consultez le site OS X Daily.
Ctrl+C	Annule la commande en cours.

Clé	Description de la fonction
Ctrl+D	Signale la fin du fichier (EOF) et quitte le niveau actuel du shell. Si aucune transaction n'est active, quitte le shell. Dans le cas d'une transaction active, annule la transaction.

Commandes de base de données Shell

Commande	Description de la fonction
`help`	Affiche les informations d'aide.
`begin`	Commence une transaction.
`start transaction`	
`commit`	Enregistre votre transaction dans le journal du registre.
`abort`	Arrête votre transaction et rejette toutes les modifications que vous avez apportées.
`exit`	Quitte la coque.
`quit`	

> **Note**
>
> Toutes les commandes shell QLDB ne sont pas sensibles à la casse.

Exécuter des relevés individuels

À l'exception des commandes de base de données et des méta-commandes du shell répertoriées dans README.md, le shell interprète chaque commande que vous entrez comme une instruction PartiQL distincte. Par défaut, le shell active le`auto-commit` mode. Ce mode est configurable.

Dans ceauto-commit mode, le shell exécute implicitement chaque instruction dans sa propre transaction et valide automatiquement la transaction si aucune erreur n'est détectée. Cela signifie que vous n'avez pas à exécuterstart transaction (oubegin)commit manuellement chaque fois que vous exécutez une instruction.

Gestion des transactions

Le shell QLDB vous permet également de contrôler manuellement les transactions. Vous pouvez exécuter plusieurs instructions au sein d'une transaction de manière interactive ou non interactive en regroupant les commandes et les instructions de manière séquentielle.

Transactions interactives

Pour exécuter une transaction interactive, procédez comme suit.

1. Pour commencer une transaction, entrez labegin commande.

   ```
   qldb> begin
   ```

   Une fois que vous avez commencé une transaction, le shell affiche l'invite de commande suivante.

   ```
   qldb *>
   ```

2. Ensuite, chaque relevé que vous saisissez s'inscrit dans la même transaction.
   - Par exemple, vous pouvez exécuter une instruction unique comme suit.

     ```
     qldb *> SELECT * FROM Vehicle WHERE VIN = '1N4AL11D75C109151'
     ```

     Une fois que vous avez appuyéEnter, le shell affiche les résultats de l'instruction.
   - Vous pouvez également saisir plusieurs instructions ou commandes séparées par un point-virgule (`;`) comme suit.

     ```
     qldb *> SELECT * FROM Vehicle WHERE VIN = '1N4AL11D75C109151'; commit
     ```

3. Pour terminer la transaction, saisissez l'une des commandes suivantes.
   - Entrez lacommit commande pour valider votre transaction dans le journal du registre.

```
qldb *> commit
```

- Entrez la`abort` commande pour arrêter votre transaction et rejeter toutes les modifications que vous avez apportées.

```
qldb *> abort
transaction was aborted
```

Limite du délai d'expiration des transactions

Une transaction interactive respecte le délai d'expiration des transactions de QLDB. Si vous ne validez pas de transaction dans les 30 secondes suivant son démarrage, QLDB fait automatiquement expirer la transaction et rejette toutes les modifications apportées au cours de la transaction.

Ensuite, au lieu d'afficher les résultats de l'instruction, le shell affiche un message d'erreur d'expiration et revient à l'invite de commande normale. Pour réessayer, vous devez saisir à nouveau la`begin` commande pour démarrer une nouvelle transaction.

```
transaction failed after 1 attempts, last error: communication failure:
 Transaction 2UMpiJ5hh7WLjVgEiML0oO has expired
```

Transactions non interactives

Vous pouvez exécuter une transaction complète avec plusieurs instructions en regroupant les commandes et les instructions de manière séquentielle comme suit.

```
qldb> begin; SELECT * FROM Vehicle WHERE VIN = '1N4AL11D75C109151'; SELECT * FROM
 Person p, DriversLicense l WHERE p.GovId = l.LicenseNumber; commit
```

Vous devez séparer chaque commande et chaque instruction par un séparateur point-virgule (`;`). Si une instruction de la transaction n'est pas valide, le shell rejette automatiquement la transaction. Le shell ne passe pas à l'exécution des instructions suivantes que vous avez saisies.

Vous pouvez également configurer plusieurs transactions.

```
qldb> begin; statement1; commit; begin; statement2; statement3; commit
```

Comme dans l'exemple précédent, si une transaction échoue, le shell ne procède à aucune transaction ou déclaration ultérieure que vous avez saisie.

Si vous ne mettez pas fin à une transaction, le shell passe en mode interactif et vous invite à entrer la commande ou l'instruction suivante.

```
qldb> begin; statement1; commit; begin
qldb *>
```

Sortir de la coque

Pour quitter la sessionqldb shell en cours, entrez laquit commandeexit or, ou utilisez le raccourci clavierCtrl +D lorsque le shell ne figure pas dans une transaction.

```
qldb> exit
$
```

```
qldb> quit
$
```

Exemple

Pour plus d'informations sur l'écriture d'instructions PartiQL dans QLDB, consultez leRéférence Amazon QLDB.

Example

L'exemple suivant montre une séquence courante de commandes de base.

> **Note**
>
> Le shell QLDB exécute chaque instruction PartiQL de cet exemple dans sa propre transaction.
> Cet exemple suppose que le registre existetest-ledger déjà et qu'il est actif.

```
$ qldb --ledger test-ledger --region us-east-1
```

```
qldb> CREATE TABLE TestTable
qldb> INSERT INTO TestTable `{"Name": "John Doe"}`
qldb> SELECT * FROM TestTable
qldb> DROP TABLE TestTable
qldb> exit
```

Accès à Amazon QLDB à l'aide de l'API

Vous pouvez utiliser le AWS Management Console et le AWS Command Line Interface (AWS CLI) pour travailler de manière interactive avec Amazon QLDB. Toutefois, pour tirer le meilleur parti de QLDB, vous pouvez écrire du code d'application à l'aide d'un pilote QLDB ou d' AWS un SDK pour interagir avec votre registre à l'aide des API.

Le pilote permet à votre application d'interagir avec QLDB à l'aide de l'API de données transactionnelles. Le AWS SDK prend en charge l'interaction avec l'API de gestion des ressources QLDB. Pour plus d'informations sur ces API, consultez leRéférence d'API Amazon QLDB.

Le pilote prend en charge QLDB en Java, .NET, Go , Node.js et Python. Pour démarrer rapidement avec ces langages, consultez Démarrage QLDB.

Avant de pouvoir utiliser un pilote QLDB ou AWS un SDK dans votre application, vous devez accorder un accès programmatique. Pour plus d'informations, voir Octroi d’un accès par programmation.

Mise en route avec la console Amazon QLDB

Ce didacticiel vous explique les étapes à suivre pour créer votre premier registre Amazon QLDB et le remplir avec des tables et des exemples de données. L'exemple de registre que vous créez dans ce scénario est une base de données pour une application du ministère des véhicules automobiles (DMV) qui permet de suivre l'historique complet des immatriculations de véhicules.

L'historique d'un actif est un cas d'utilisation courant pour QLDB car il implique une variété de scénarios et d'opérations qui mettent en évidence l'utilité d'une base de données grand livre. Avec QLDB, vous pouvez accéder directement à l'historique complet des modifications apportées à vos données, l'interroger et le vérifier dans une base de données orientée document qui prend en charge des fonctionnalités de requête de type SQL.

Au cours de ce didacticiel, les rubriques suivantes expliquent comment ajouter des immatriculations de véhicules, les modifier et consulter l'historique des modifications apportées à ces immatriculations. Ce guide explique également comment vérifier un document d'enregistrement de manière cryptographique et conclut en nettoyant les ressources et en supprimant le registre d'échantillons.

Chaque étape du didacticiel contient des instructions d'utilisation duAWS Management Console.

Rubriques

Prérequis et considérations relatives au didacticiel

Avant de commencer ce didacticiel Amazon QLDB, assurez-vous de respecter les conditions préalables suivantes :

1. Suivez les instructions deAWS configuration dansAccès à Amazon QLDB, si vous ne l'avez pas déjà fait. Ces étapes incluent l'inscriptionAWS et la création d'un utilisateur administratif.
2. Suivez les instructions de laConfigurer les autorisations section pour configurer les autorisations IAM pour vos ressources QLDB. Pour compléter toutes les étapes de ce didacticiel, vous devez disposer d'un accès administratif complet à vos ressources de registre via leAWS Management Console.

   > **Note**
   >
   > Si vous êtes déjà connecté en tant qu'utilisateur disposant de droitsAWS administratifs, vous pouvez ignorer cette étape.

3. (Facultatif) QLDB chiffre les données au repos à l'aide d'une cléAWS Key Management Service (AWS KMS). Vous pouvez choisir l'un des types suivantsAWS KMS keys :

   - AWSclé KMS détenue — Utilisez une clé KMS détenue et gérée parAWS en votre nom. Il s'agit de l'option par défaut qui ne nécessite aucune configuration supplémentaire.
   - Clé KMS gérée par le client : utilisez une clé KMS de chiffrement symétrique dans votre compte que vous créez, possédez et gérez. QLDB ne prend pas en charge les clés asymétriques.

     Cette option nécessite que vous créiez une clé KMS ou que vous utilisiez une clé existante dans votre compte. Pour obtenir des instructions sur la création d'une clé gérée par le client, consultez la section Création de clés KMS à chiffrement symétrique dans le Guide duAWS Key Management Service développeur.

     Vous pouvez spécifier une clé KMS gérée par le client en utilisant un ID, un alias ou un nom de ressource Name (ARN). Pour en savoir plus, consultez la section Identificateurs clés (KeyId) dans le Guide duAWS Key Management Service développeur.

     > **Note**
     >
     > Les clés inter-régions ne sont pas prises en charge. La clé KMS spécifiée doit êtreRégion AWS identique à votre registre.

Configurer les autorisations

Au cours de cette étape, vous configurez des autorisations d'accès complètes via la console pour toutes les ressources QLDB de votreCompte AWS. Pour accorder ces autorisations rapidement, utilisez la politiqueAWS gérée d'AmazonQLDBConsoleFullAccess.

Pour activer l'accès, ajoutez des autorisations à vos utilisateurs, groupes ou rôles :

- Utilisateurs et groupes dans AWS IAM Identity Center :

  Créez un jeu d'autorisations. Suivez les instructions de la rubrique Création d'un jeu d'autorisations du Guide de l'utilisateur AWS IAM Identity Center.

- Utilisateurs gérés dans IAM par un fournisseur d'identité :

  Créez un rôle pour la fédération d'identité. Pour plus d'informations, voir la rubrique Création d'un rôle pour un fournisseur d'identité tiers (fédération) du Guide de l'utilisateur IAM.

- Utilisateurs IAM :
  - Créez un rôle que votre utilisateur peut assumer. Suivez les instructions de la rubrique Création d'un rôle pour un utilisateur IAM du Guide de l'utilisateur IAM.
  - (Non recommandé) Attachez une politique directement à un utilisateur ou ajoutez un utilisateur à un groupe d'utilisateurs. Suivez les instructions de la rubrique Ajout d'autorisations à un utilisateur (console) du Guide de l'utilisateur IAM.

> **Important**
>
> Dans le cadre de ce didacticiel, vous vous accordez un accès administratif complet à toutes les ressources QLDB. Pour les cas d'utilisation en production, suivez toutefois la meilleure pratique de sécurité du moindre privilège, principe selon lequel il ne faut accorder que les autorisations requises pour une seule tâche. Pour obtenir des exemples, consultez Exemples de politiques basées sur l'identité pour Amazon QLDB.

Pour créer un registre nommé`vehicle-registration`, passez àÉtape 1 : Créer un nouveau registre.

Étape 1 : Créer un nouveau registre

Au cours de cette étape, vous allez créer un nouveau registre Amazon QLDB nommévehicle-registration. Ensuite, vous confirmez que le statut du registre est Actif. Vous pouvez également vérifier toutes les balises que vous avez ajoutées au registre.

Lorsque vous créez un registre, la protection de la suppression est activée par défaut. La protection contre la suppression est une fonctionnalité de QLDB qui empêche la suppression des registres par un utilisateur. Vous pouvez désactiver la protection contre la suppression lorsque vous créez un registre à l'aide de l'API QLDB ou duAWS Command Line Interface (AWS CLI).

Pour créer un nouveau registre

1. Connectez-vous au et ouvrezAWS Management Console la console Amazon QLDB à l'adresse https://console.aws.amazon.com/qldb.
2. Dans le panneau de navigation, choisissez Mise en route.
3. Sur la page Créez votre première fiche comptable, choisissez Créer un livre.
4. Sur la page Create Ledger (Créer un registre), procédez de la manière suivante :

   - Informations sur le registre — Le nom du registre doit être prérempli avec**vehicle-registration**.
   - Mode autorisations — Mode d'autorisations à attribuer au registre. Choisissez l'une des options suivantes :
     - Tout autoriser : un mode d'autorisations hérité qui permet le contrôle d'accès avec une granularité au niveau de l'API pour les registres.

       Ce mode permet aux utilisateurs qui possèdent l'autorisation d'API SendCommand pour ce registre d'exécuter toutes les commandes PartiQL (par conséquent, ALLOW_ALL) sur toutes les tables du registre spécifié. Ce mode ignore les politiques d'autorisations IAM au niveau de la table ou de la commande que vous créez pour le registre.
     - Standard : (Recommandé) un mode d'autorisations qui permet le contrôle d'accès avec une granularité plus fine pour les registres, les tables et les commandes PartiQL. Nous vous recommandons vivement d'utiliser ce mode d'autorisations pour optimiser la sécurité des données de votre registre.

       Par défaut, ce mode refuse toutes les demandes d'exécuter des commandes PartiQL sur les tables de ce registre. Pour autoriser les commandes PartiQL, vous devez créer

des politiques d'autorisations IAM pour des ressources de table et des actions PartiQL spécifiques, en plus de l'autorisation d'`SendCommand`API pour le registre. Pour plus d'informations, consultez Commencer à utiliser le mode d'autorisation standard dans Amazon QLDB.

- Chiffrer les données au repos — La clé dansAWS Key Management Service (AWS KMS) à utiliser pour le chiffrement des données au repos. Choisissez l'une des options suivantes :
  - Utiliser une clé KMSAWS détenue : utilisez une clé KMS détenue et gérée parAWS en votre nom. Il s'agit de l'option par défaut qui ne nécessite aucune configuration supplémentaire.
  - Choisissez uneAWS KMS clé différente : utilisez une clé KMS de chiffrement symétrique dans votre compte que vous créez, possédez et gérez.

    Pour créer une nouvelle clé à l'aide de laAWS KMS console, choisissez Créer uneAWS KMS clé. Pour plus d'informations, consultez Création de clés KMS de chiffrement symétriques dans le Guide du développeur AWS Key Management Service.

    Pour utiliser une clé KMS existante, choisissez-en une dans la liste déroulante ou spécifiez un ARN de clé KMS.
- Etiquettes — (Facultatif) Ajoutez des métadonnées au registre en associant les balises sous forme de paires clé-valeur. Vous pouvez ajouter des étiquettes à votre registre pour mieux les organiser et les identifier. Pour plus d'informations, veuillez consulter Balisage des ressources Amazon QLDB.

  Choisissez Ajouter une balise, puis entrez les paires clé-valeur appropriées.

5. Lorsque les paramètres vous conviennent, choisissez Create ledger.

> **Note**
>
> Vous pouvez accéder à votre registre QLDB lorsque son statut devient Actif. Cela peut prendre plusieurs minutes.

6. Dans la liste des livres, recherchez`vehicle-registration` et confirmez que le statut du registre est actif.
7. (Facultatif) Choisissez le nom du`vehicle-registration` registre. Sur la page des détails du registre d'immatriculation des véhicules, vérifiez que toutes les étiquettes que vous avez ajoutées au registre apparaissent sur la carte Tags. Vous pouvez également modifier vos balises de registre à l'aide de cette page de console.

Pour créer des tables dans le`vehicle-registration` grand livre, passez àÉtape 2 : Création de tables, d'index et d'exemples de données dans un registre.

Étape 2 : Création de tables, d'index et d'exemples de données dans un registre

Lorsque votre registre Amazon QLDB est actif, vous pouvez commencer à créer des tableaux contenant des données sur les véhicules, leurs propriétaires et leurs informations d'immatriculation. Après avoir créé les tables et les index, vous pouvez les charger avec des données.

Au cours de cette étape, vous allez créer quatre tables dans le`vehicle-registration` registre :

- `VehicleRegistration`
- `Vehicle`
- `Person`
- `DriversLicense`

Vous pouvez également créer les index suivants.

Nom de la table	Champ
`VehicleRegistration`	`VIN`
`VehicleRegistration`	`LicensePlateNumber`
`Vehicle`	`VIN`
`Person`	`GovId`
`DriversLicense`	`LicensePlateNumber`
`DriversLicense`	`PersonId`

Vous pouvez utiliser la console QLDB pour créer automatiquement ces tables avec des index et les charger avec des exemples de données. Vous pouvez également utiliser l'éditeur PartiQL sur la console pour exécuter manuellement chaque instruction PartiQL step-by-step.

Option automatique

Pour créer des tables, des index et des exemples de données

1. Ouvrez la console Amazon QLDB à l'adresse https://console.aws.amazon.com/qldb.
2. Dans le panneau de navigation, choisissez Mise en route.
3. Sous l'option Automatique de la carte de données d'application Exemple, choisissez`vehicle-registration` dans la liste des registres.
4. Choisissez Charger des exemples de données.

   Si l'opération se termine correctement, la console affiche le message Sample data loaded.

   Ce script exécute toutes les instructions dans une transaction unique. Si une partie de la transaction échoue, chaque instruction est annulée et un message d'erreur approprié s'affiche. Vous pouvez recommencer l'opération après avoir résolu tout problème.

   > **Note**
   >
   > - L'une des causes possibles de l'échec d'une transaction est la tentative de création de tables dupliquées. Votre demande de chargement d'exemples de données échouera si l'un des noms de table suivants existe déjà dans votre registre :`VehicleRegistration`Vehicle,`Person`, et`DriversLicense`.
   >
   >   Essayez plutôt de charger ces exemples de données dans un registre vide.
   > - Ce script exécute des`INSERT` instructions paramétrées. Ainsi, ces instructions partiQL sont enregistrées dans vos blocs de journal avec des paramètres de liaison au lieu de données littérales. Par exemple, vous pouvez voir la déclaration suivante dans un bloc de journal, où le point d'interrogation (?) est un espace réservé variable pour le contenu du document.
   >
   >   ```
   >   INSERT INTO Vehicle ?
   >   ```

Option manuelle

Vous insérez des documents dans`VehicleRegistration` un`PrimaryOwner` champ vide et dans`DriversLicense` un`PersonId` champ vide. Vous renseignez ensuite ces champs avec le document attribué par le système`id` à partir de la`Person` table.

> **Tip**
>
> Il est recommandé d'utiliser ce champ `id` métadonnées du document comme clé étrangère. Pour plus d'informations, veuillez consulter Interroger les métadonnées d'un document.

Pour créer des tables, des index et des exemples de données

1. Ouvrez la console Amazon QLDB à l'adresse https://console.aws.amazon.com/qldb.
2. Dans le panneau de navigation, choisissez PartiQL Editor.
3. Choisissez le`vehicle-registration` registre.
4. Commencez par créer quatre tables. QLDB prend en charge le contenu ouvert et n'applique pas de schéma. Vous ne devez donc pas spécifier d'attributs ou de types de données.

   Dans la fenêtre de l'éditeur de requêtes, entrez l'instruction suivante, puis choisissez Exécuter. Pour exécuter l'instruction, vous pouvez également utiliser le raccourci clavierCtrl +Enter pour Windows ouCmd +Return pour macOS. Pour plus de raccourcis clavier, reportez-vous à la sectionRaccourcis clavier de l'éditeur PartiQL.

   ```
   CREATE TABLE VehicleRegistration
   ```

   Répétez cette étape pour chacune des étapes suivantes.

   ```
   CREATE TABLE Vehicle
   ```

   ```
   CREATE TABLE Person
   ```

   ```
   CREATE TABLE DriversLicense
   ```

5. Créez ensuite des index qui optimisent les performances des requêtes pour chaque table.

> **Important**
>
> QLDB a besoin d'un index pour rechercher efficacement un document. Sans index, QLDB doit effectuer une analyse complète de la table lors de la lecture de documents. Cela peut entraîner des problèmes de performances sur des tables de grande taille, notamment des conflits de simultanéité et des délais de transaction.
> Pour éviter de scanner des tables, vous devez exécuter des instructions avec une clause de`WHERE` prédicat à l'aide d'un opérateur d'égalité (`=`ou`IN`) sur un champ indexé ou un identifiant de document. Pour plus d'informations, veuillez consulter Optimisation des performances des données.

Dans la fenêtre de l'éditeur de requêtes, entrez l'instruction suivante, puis choisissez Exécuter.

```
CREATE INDEX ON VehicleRegistration (VIN)
```

Répétez cette étape pour les étapes suivantes.

```
CREATE INDEX ON VehicleRegistration (LicensePlateNumber)
```

```
CREATE INDEX ON Vehicle (VIN)
```

```
CREATE INDEX ON Person (GovId)
```

```
CREATE INDEX ON DriversLicense (LicensePlateNumber)
```

```
CREATE INDEX ON DriversLicense (PersonId)
```

6. Après avoir créé vos index, vous pouvez commencer à charger des données dans vos tables. Au cours de cette étape, insérez des documents dans le`Person` tableau contenant des informations personnelles sur les propriétaires des véhicules suivis par le registre.

   Dans la fenêtre de l'éditeur de requêtes, entrez l'instruction suivante, puis choisissez Exécuter.

```
INSERT INTO Person
<< {
```

```
    'FirstName' : 'Raul',
    'LastName' : 'Lewis',
    'DOB' : `1963-08-19T`,
    'GovId' : 'LEWISR261LL',
    'GovIdType' : 'Driver License',
    'Address' : '1719 University Street, Seattle, WA, 98109'
},
{
    'FirstName' : 'Brent',
    'LastName' : 'Logan',
    'DOB' : `1967-07-03T`,
    'GovId' : 'LOGANB486CG',
    'GovIdType' : 'Driver License',
    'Address' : '43 Stockert Hollow Road, Everett, WA, 98203'
},
{
    'FirstName' : 'Alexis',
    'LastName' : 'Pena',
    'DOB' : `1974-02-10T`,
    'GovId' : '744 849 301',
    'GovIdType' : 'SSN',
    'Address' : '4058 Melrose Street, Spokane Valley, WA, 99206'
},
{
    'FirstName' : 'Melvin',
    'LastName' : 'Parker',
    'DOB' : `1976-05-22T`,
    'GovId' : 'P626-168-229-765',
    'GovIdType' : 'Passport',
    'Address' : '4362 Ryder Avenue, Seattle, WA, 98101'
},
{
    'FirstName' : 'Salvatore',
    'LastName' : 'Spencer',
    'DOB' : `1997-11-15T`,
    'GovId' : 'S152-780-97-415-0',
    'GovIdType' : 'Passport',
    'Address' : '4450 Honeysuckle Lane, Seattle, WA, 98101'
} >>
```

7. Remplissez ensuite le`DriversLicense` tableau avec des documents contenant les informations relatives au permis de conduire de chaque propriétaire de véhicule.

   Dans la fenêtre de l'éditeur de requêtes, entrez l'instruction suivante, puis choisissez Exécuter.

```
INSERT INTO DriversLicense
<< {
    'LicensePlateNumber' : 'LEWISR261LL',
    'LicenseType' : 'Learner',
    'ValidFromDate' : `2016-12-20T`,
    'ValidToDate' : `2020-11-15T`,
    'PersonId' : ''
},
{
    'LicensePlateNumber' : 'LOGANB486CG',
    'LicenseType' : 'Probationary',
    'ValidFromDate' : `2016-04-06T`,
    'ValidToDate' : `2020-11-15T`,
    'PersonId' : ''
},
{
    'LicensePlateNumber' : '744 849 301',
    'LicenseType' : 'Full',
    'ValidFromDate' : `2017-12-06T`,
    'ValidToDate' : `2022-10-15T`,
    'PersonId' : ''
},
{
    'LicensePlateNumber' : 'P626-168-229-765',
    'LicenseType' : 'Learner',
    'ValidFromDate' : `2017-08-16T`,
    'ValidToDate' : `2021-11-15T`,
    'PersonId' : ''
},
{
    'LicensePlateNumber' : 'S152-780-97-415-0',
    'LicenseType' : 'Probationary',
    'ValidFromDate' : `2015-08-15T`,
    'ValidToDate' : `2021-08-21T`,
    'PersonId' : ''
} >>
```

8. Maintenant, remplissez le`VehicleRegistration` tableau avec les documents d'immatriculation du véhicule. Ces documents incluent une`Owners` structure imbriquée qui enregistre les propriétaires principaux et secondaires.

   Dans la fenêtre de l'éditeur de requêtes, entrez l'instruction suivante, puis choisissez Exécuter.

```
INSERT INTO VehicleRegistration
<< {
    'VIN' : '1N4AL11D75C109151',
    'LicensePlateNumber' : 'LEWISR261LL',
    'State' : 'WA',
    'City' : 'Seattle',
    'PendingPenaltyTicketAmount' : 90.25,
    'ValidFromDate' : `2017-08-21T`,
    'ValidToDate' : `2020-05-11T`,
    'Owners' : {
        'PrimaryOwner' : { 'PersonId': '' },
        'SecondaryOwners' : []
    }
},
{
    'VIN' : 'KM8SRDHF6EU074761',
    'LicensePlateNumber' : 'CA762X',
    'State' : 'WA',
    'City' : 'Kent',
    'PendingPenaltyTicketAmount' : 130.75,
    'ValidFromDate' : `2017-09-14T`,
    'ValidToDate' : `2020-06-25T`,
    'Owners' : {
        'PrimaryOwner' : { 'PersonId': '' },
        'SecondaryOwners' : []
    }
},
{
    'VIN' : '3HGGK5G53FM761765',
    'LicensePlateNumber' : 'CD820Z',
    'State' : 'WA',
    'City' : 'Everett',
    'PendingPenaltyTicketAmount' : 442.30,
    'ValidFromDate' : `2011-03-17T`,
    'ValidToDate' : `2021-03-24T`,
    'Owners' : {
        'PrimaryOwner' : { 'PersonId': '' },
        'SecondaryOwners' : []
    }
},
{
    'VIN' : '1HVBBAANXWH544237',
    'LicensePlateNumber' : 'LS477D',
```

```
    'State' : 'WA',
    'City' : 'Tacoma',
    'PendingPenaltyTicketAmount' : 42.20,
    'ValidFromDate' : `2011-10-26T`,
    'ValidToDate' : `2023-09-25T`,
    'Owners' : {
        'PrimaryOwner' : { 'PersonId': '' },
        'SecondaryOwners' : []
    }
},
{
    'VIN' : '1C4RJFAG0FC625797',
    'LicensePlateNumber' : 'TH393F',
    'State' : 'WA',
    'City' : 'Olympia',
    'PendingPenaltyTicketAmount' : 30.45,
    'ValidFromDate' : `2013-09-02T`,
    'ValidToDate' : `2024-03-19T`,
    'Owners' : {
        'PrimaryOwner' : { 'PersonId': '' },
        'SecondaryOwners' : []
    }
} >>
```

9. Enfin, remplissez le`Vehicle` tableau avec les documents décrivant les véhicules enregistrés dans votre registre.

   Dans la fenêtre de l'éditeur de requêtes, entrez l'instruction suivante, puis choisissez Exécuter.

```
INSERT INTO Vehicle
<< {
    'VIN' : '1N4AL11D75C109151',
    'Type' : 'Sedan',
    'Year' : 2011,
    'Make' : 'Audi',
    'Model' : 'A5',
    'Color' : 'Silver'
},
{
    'VIN' : 'KM8SRDHF6EU074761',
    'Type' : 'Sedan',
    'Year' : 2015,
    'Make' : 'Tesla',
```

```
    'Model' : 'Model S',
    'Color' : 'Blue'
},
{
    'VIN' : '3HGGK5G53FM761765',
    'Type' : 'Motorcycle',
    'Year' : 2011,
    'Make' : 'Ducati',
    'Model' : 'Monster 1200',
    'Color' : 'Yellow'
},
{
    'VIN' : '1HVBBAANXWH544237',
    'Type' : 'Semi',
    'Year' : 2009,
    'Make' : 'Ford',
    'Model' : 'F 150',
    'Color' : 'Black'
},
{
    'VIN' : '1C4RJFAG0FC625797',
    'Type' : 'Sedan',
    'Year' : 2019,
    'Make' : 'Mercedes',
    'Model' : 'CLK 350',
    'Color' : 'White'
} >>
```

Vous pouvez ensuite utiliser`SELECT` des instructions pour lire les données des tables du`vehicle-registration` registre. Passez à Étape 3 : Interrogation des tables dans un registre.

Étape 3 : Interrogation des tables dans un registre

Après avoir créé des tableaux dans un registre Amazon QLDB et les avoir chargés de données, vous pouvez exécuter des requêtes pour vérifier les données d'immatriculation du véhicule que vous venez d'insérer. QLDB utilise PartiQL comme langage de requête et Amazon Ion comme modèle de données orienté document.

PartiQL est un langage de requête open source compatible SQL qui a été étendu pour fonctionner avec Ion. Avec PartiQL, vous pouvez insérer, interroger et gérer vos données à l'aide d'opérateurs SQL familiers. Amazon Ion est un sur-ensemble de JSON. Ion est un format de données open source

basé sur des documents qui vous permet de stocker et de traiter des données structurées, semi-structurées et imbriquées.

Au cours de cette étape, vous utilisez des`SELECT` instructions pour lire les données des tables du`vehicle-registration` registre.

> **Warning**
>
> Lorsque vous exécutez une requête dans QLDB sans recherche indexée, elle appelle une analyse complète de la table. PartiQL prend en charge de telles requêtes car il est compatible avec SQL. Toutefois, n'exécutez pas d'analyses de tables pour des cas d'utilisation en production dans QLDB. Les analyses de tables peuvent entraîner des problèmes de performances sur des tables de grande taille, notamment des conflits de simultanéité et des délais de transaction.
>
> Pour éviter de scanner des tables, vous devez exécuter des instructions avec une clause de`WHERE` prédicat à l'aide d'un opérateur d'égalité sur un champ indexé ou un identifiant de document, par exemple,`WHERE indexedField = 123` ou`WHERE indexedField IN (456, 789)`. Pour plus d'informations, veuillez consulter Optimisation des performances des données.

Pour interroger les tables

1. Ouvrez la console Amazon QLDB à l'adresse https://console.aws.amazon.com/qldb.
2. Dans le panneau de navigation, choisissez PartiQL Editor.
3. Choisissez le`vehicle-registration` registre.
4. Dans la fenêtre de l'éditeur de requêtes, entrez l'instruction suivante pour rechercher dans la`Vehicle` table un numéro d'identification de véhicule (VIN) particulier que vous avez ajouté au registre, puis choisissez Exécuter.

   Pour exécuter l'instruction, vous pouvez également utiliser le raccourci clavierCtrl +Enter pour Windows ouCmd +Return pour macOS. Pour plus de raccourcis clavier, reportez-vous à la sectionRaccourcis clavier de l'éditeur PartiQL.

   ```
   SELECT * FROM Vehicle AS v
   WHERE v.VIN = '1N4AL11D75C109151'
   ```

5. Vous pouvez écrire des requêtes de jointure internes. Cet exemple de requêteVehicle jointVehicleRegistration et renvoie les informations d'immatriculation ainsi que les attributs du véhicule enregistré pour une donnéeVIN.

   Entrez l'instruction suivante, puis choisissez Exécuter.

   ```
   SELECT v.VIN, r.LicensePlateNumber, r.State, r.City, r.Owners
   FROM Vehicle AS v, VehicleRegistration AS r
   WHERE v.VIN = '1N4AL11D75C109151'
   AND v.VIN = r.VIN
   ```

   Vous pouvez également joindre lesDriversLicense tablesPerson et pour voir les attributs liés aux conducteurs qui ont été ajoutés au registre.

   Répétez cette étape pour les étapes suivantes.

   ```
   SELECT * FROM Person AS p, DriversLicense AS l
   WHERE p.GovId = l.LicensePlateNumber
   ```

Pour en savoir plus sur la modification de documents dans les tableaux duvehicle-registration grand livre, reportez-vous à la sectionÉtape 4 : Modifier les documents dans un registre.

Étape 4 : Modifier les documents dans un registre

Maintenant que vous avez des données sur lesquelles travailler, vous pouvez commencer à apporter des modifications aux documents duvehicle-registration registre dans Amazon QLDB. Prenons l'exemple de l'Audi A5 avec VIN1N4AL11D75C109151. Cette voiture appartient initialement à un pilote nommé Raul Lewis à Seattle, dans l'État de Washington.

Supposons que Raul vende la voiture à un habitant d'Everett, dans l'État de Washington, nommé Brent Logan. Ensuite, Brent et Alexis Pena décident de se marier. Brent souhaite ajouter Alexis en tant que propriétaire secondaire sur l'enregistrement. Au cours de cette étape, les instructions du langage de manipulation de données (DML) suivantes montrent comment apporter les modifications appropriées à votre registre pour refléter ces événements.

> **Tip**
>
> La meilleure pratique consiste à utiliser une clé étrangère attribuée par`id` le système à un document. Bien que vous puissiez définir des champs destinés à être des identifiants uniques (par exemple, le VIN d'un véhicule), le véritable identifiant unique d'un document est le sien`id`. Ce champ est inclus dans les métadonnées du document, que vous pouvez interroger dans la vue validée (la vue définie par le système d'une table).
> Pour de plus d'informations sur les vues dans QLDB, consultezConcepts de base. Pour en savoir plus sur les métadonnées, consultezInterroger les métadonnées d'un document.

Pour modifier des documents

1. Ouvrez la console Amazon QLDB à l'adresse https://console.aws.amazon.com/qldb.
2. Dans le panneau de navigation, choisissez PartiQL Editor.
3. Choisissez le`vehicle-registration` registre.

   > **Note**
   >
   > Si vous configurez votre registre à l'aide de la fonction de chargement automatique des exemples de données de la console, passez directement à l'étape 6.

4. Si vous avez exécuté manuellement`INSERT` des instructions pour charger les exemples de données, procédez comme suit.

   Pour enregistrer Raul en tant que propriétaire de ce véhicule, commencez par rechercher le document attribué par le système`id` dans le`Person` tableau. Ce champ est inclus dans les métadonnées du document, que vous pouvez interroger dans la vue définie par le système de la table, appelée vue validée.

   Dans la fenêtre de l'éditeur de requêtes, entrez l'instruction suivante, puis choisissez Exécuter.

   ```
   SELECT metadata.id FROM _ql_committed_Person AS p
   WHERE p.data.FirstName = 'Raul' and p.data.LastName = 'Lewis'
   ```

   Le préfixe`_ql_committed_` est un préfixe réservé qui signifie que vous souhaitez interroger la vue validée de la`Person` table. Dans cette vue, vos données sont imbriquées dans le`data` champ et les métadonnées sont imbriquées dans le`metadata` champ.

5. Maintenant, utilisez-le`id` dans une`UPDATE` instruction pour modifier le document approprié dans le`VehicleRegistration` tableau. Entrez l'instruction suivante, puis choisissez Exécuter.

```
UPDATE VehicleRegistration AS r
SET r.Owners.PrimaryOwner.PersonId = '294jJ3YUoH1IEEm8GSabOs' --replace with your
 id
WHERE r.VIN = '1N4AL11D75C109151'
```

   Confirmez que vous avez modifié le`Owners` champ en émettant cette déclaration.

```
SELECT r.Owners FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
```

6. Pour transférer la propriété du véhicule à Brent, dans la ville d'Everett, trouvez d'abord le sien dans le`Person` tableau à l'aide`id` de la déclaration suivante.

```
SELECT metadata.id FROM _ql_committed_Person AS p
WHERE p.data.FirstName = 'Brent' and p.data.LastName = 'Logan'
```

   Ensuite, utilisez-le`id` pour mettre à jour le`PrimaryOwner` et le`City` dans le`VehicleRegistration` tableau.

```
UPDATE VehicleRegistration AS r
SET r.Owners.PrimaryOwner.PersonId = '7NmE8YLPbXc0IqesJy1rpR', --replace with your
 id
    r.City = 'Everett'
WHERE r.VIN = '1N4AL11D75C109151'
```

   Confirmez que vous avez modifié les`City` champs`PrimaryOwner` et en émettant cette déclaration.

```
SELECT r.Owners.PrimaryOwner, r.City
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
```

7. Pour ajouter Alexis en tant que propriétaire secondaire de la voiture, trouvez-la`Person id`.

```
SELECT metadata.id FROM _ql_committed_Person AS p
WHERE p.data.FirstName = 'Alexis' and p.data.LastName = 'Pena'
```

Ensuite, insérez-le`id` dans la`SecondaryOwners` liste à l'aide de l'instruction DML FROM-INSERT suivante.

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
INSERT INTO r.Owners.SecondaryOwners
    VALUE { 'PersonId' : '5Ufgdlnj06gF5CWc0Iu64s' } --replace with your id
```

Confirmez que vous avez modifié`SecondaryOwners` en émettant cette déclaration.

```
SELECT r.Owners.SecondaryOwners FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
```

Pour consulter ces modifications dans le`vehicle-registration` registre, reportez-vous à la sectionÉtape 5 : Afficher l'historique des révisions d'un document.

Étape 5 : Afficher l'historique des révisions d'un document

Après avoir modifié les données d'immatriculation de la voiture avec le VIN `1N4AL11D75C109151`, vous pouvez consulter l'historique de tous ses propriétaires enregistrés et tout autre champ mis à jour. Vous pouvez voir toutes les révisions d'un document que vous avez insérées, mises à jour et supprimées en interrogeant la version intégréeFonction historique.

La fonction d'historique renvoie les révisions à partir de la vue validée de votre table, qui inclut à la fois les données de votre application et les métadonnées associées. Les métadonnées indiquent exactement quand chaque révision a été effectuée, dans quel ordre et quelle transaction les a validées.

Au cours de cette étape, vous interrogez l'historique des révisions d'un document dans le`VehicleRegistration` tableau du`vehicle-registration` grand livre.

Pour consulter l'historique des révisions

1. Ouvrez la console Amazon QLDB à l'adresse https://console.aws.amazon.com/qldb.
2. Dans le panneau de navigation, choisissez PartiQL Editor.
3. Choisissez le`vehicle-registration` registre.

4. Pour interroger l'historique d'un document, commencez par trouver son caractère uniqueid. Outre l'interrogation de la vue validée, un autre moyen d'obtenir un documentid consiste à utiliser leBY mot clé dans la vue utilisateur par défaut du tableau. Pour en savoir plus, consultez Utilisation de la clause BY pour interroger l'ID du document.

   Dans la fenêtre de l'éditeur de requêtes, entrez l'instruction suivante, puis choisissez Exécuter.

```
SELECT r_id FROM VehicleRegistration AS r BY r_id
WHERE r.VIN = '1N4AL11D75C109151'
```

5. Vous pouvez ensuite utiliser cetteid valeur pour interroger la fonction d'historique. Entrez l'instruction suivante, puis choisissez Exécuter. Veillez à remplacer laid valeur par votre ID de document, le cas échéant.

```
SELECT h.data.VIN, h.data.City, h.data.Owners
FROM history(VehicleRegistration) AS h
WHERE h.metadata.id = 'ADR2LQq48kB9neZDupQrMm' --replace with your id
```

> **Note**
>
> Dans le cadre de ce didacticiel, cette requête d'historique renvoie toutes les révisions de l'ID du documentADR2LQq48kB9neZDupQrMm. Il est toutefois recommandé de qualifier une requête d'historique à la fois avec un identifiant de document et une plage de dates (heure de début et heure de fin).
> Dans QLDB, chaqueSELECT requête est traitée dans le cadre d'une transaction et est soumise à un délai d'expiration de transaction. Les requêtes d'historique qui incluent une heure de début et une heure de fin bénéficient d'une qualification par plage de dates. Pour plus d'informations, veuillez consulter Fonction historique.

La fonction historique renvoie les documents dans le même schéma que la vue validée. Cet exemple projette les données d'immatriculation modifiées de votre véhicule. La sortie doit ressembler à ce qui suit :

VIN	Ville	Propriétaires
"1N4AL11D 75C109151"	"Seattle"	{PrimaryOwner:{PersonId:""} ,SecondaryOwners:[]}
"1N4AL11D 75C109151"	"Seattle"	{PrimaryOwner:{PersonId:"29 4jJ3YUoH1IEEm8GSabOs"}, SecondaryOwners:[]}
"1N4AL11D 75C109151"	"Everett"	{PrimaryOwner:{PersonId:"7N mE8YLPbXc0IqesJy1rpR"}, SecondaryOwners:[]}
"1N4AL11D 75C109151"	"Everett"	{PrimaryOwner:{PersonId:"7N mE8YLPbXc0IqesJy1rpR"}, SecondaryOwners:[{PersonId: "5Ufgdlnj06gF5CWcOIu64s"}]}

> **Note**
>
> La requête d'historique peut ne pas toujours renvoyer les révisions des documents dans un ordre séquentiel.

Passez en revue la sortie et confirmez que les modifications reflètent ce que vous avez fait dansÉtape 4 : Modifier les documents dans un registre.

6. Vous pouvez ensuite inspecter les métadonnées du document pour chaque révision. Entrez l'instruction suivante, puis choisissez Exécuter. Encore une fois, veillez à remplacer `laid` valeur par votre propre numéro de document, le cas échéant.

```
SELECT VALUE h.metadata
FROM history(VehicleRegistration) AS h
WHERE h.metadata.id = 'ADR2LQq48kB9neZDupQrMm' --replace with your id
```

La sortie doit ressembler à ce qui suit :

versio	id	TxTime	TxID
0	"ADR2LQq48kB9neZDupQrMm"	2019-05-23T19:20:360d-3Z	"FMoVdWuPxJg3k466Iz4i75"
1	"ADR2LQq48kB9neZDupQrMm"	2019-05-23T21:40:199d-3Z	"KWByxe842Xw8DNHcvARPOt"
2	"ADR2LQq48kB9neZDupQrMm"	2019-05-23T21:44:432d-3Z	"EKwDOJRwbHpFvmAyJ2Kdh9"
3	"ADR2LQq48kB9neZDupQrMm"	2019-05-23T21:49:254d-3Z	"96EiZd7vCmJ6RAvOvTZ4YA"

Ces champs de métadonnées fournissent des détails sur le moment où chaque élément a été modifié et par quelle transaction. À partir de ces données, vous pouvez déduire ce qui suit :

- Le document est identifié de manière unique par son attribut systèmeid :ADR2LQq48kB9neZDupQrMm. Il s'agit d'un identifiant unique universel (UUID) représenté dans une chaîne codée en Base62.
- CelatxTime indique que la révision initiale du document (version0) a été créée à2019-05-23T19:20:360d-3Z.
- Chaque transaction suivante crée une nouvelle révision avec le même documentid, un numéro de version incrémenté et untxId et mis à jourtxTime.

Pour vérifier la révision d'un document de manière cryptographique dans levehicle-registration registre, passez àÉtape 6 : Vérifier un document dans un registre.

Étape 6 : Vérifier un document dans un registre

Avec Amazon QLDB, vous pouvez vérifier efficacement l'intégrité d'un document dans le journal de votre registre en utilisant le hachage cryptographique SHA-256. Dans cet exemple, Alexis et Brent décident de passer à un nouveau modèle en échangeant le véhicule avec VIN1N4AL11D75C109151 chez un concessionnaire automobile. Le concessionnaire commence le processus en vérifiant la propriété du véhicule auprès du bureau d'immatriculation.

Pour en savoir plus sur le fonctionnement de la vérification et du hachage cryptographique dans QLDB, consultezVérification des données dans Amazon QLDB.

Au cours de cette étape, vous vérifiez la révision d'un document dans le`vehicle-registration` registre. Tout d'abord, vous demandez un condensé, qui est renvoyé sous forme de fichier de sortie et fait office de signature de l'historique complet des modifications de votre registre. Ensuite, vous demandez une preuve pour la révision par rapport à ce condensé. À l'aide de cette preuve, l'intégrité de votre révision est vérifiée si tous les contrôles de validation sont réussis.

Pour demander un résumé

1. Ouvrez la console Amazon QLDB à l'adresse https://console.aws.amazon.com/qldb.
2. Dans le panneau de navigation, choisissez Ledgers.
3. Dans la liste des livres, sélectionnez`vehicle-registration`.
4. Choisissez Get Digest. La boîte de dialogue Obtenir le résumé affiche les informations suivantes sur le condensé :

   - Résumé : valeur de hachage SHA-256 du condensé que vous avez demandé.
   - Adresse du résumé : dernier emplacement du bloc dans le journal couvert par le résumé que vous avez demandé. Une adresse comporte les deux champs suivants :
     - `strandId`— L'identifiant unique du volet du journal qui contient le bloc.
     - `sequenceNo`— Numéro d'index qui indique l'emplacement du bloc dans le brin.
   - Registre : nom du registre pour lequel vous avez demandé un résumé.
   - Date : horodatage auquel vous avez demandé le résumé.
5. Vérifiez les informations de synthèse. Ensuite, choisissez Save (Enregistrer). Vous pouvez conserver le nom de fichier par défaut ou saisir un nouveau nom.

   Cette étape enregistre un fichier texte brut dont le contenu est au format Amazon Ion. Le fichier porte l'extension de nom de fichier de`.ion.txt` et contient toutes les informations de synthèse répertoriées dans la boîte de dialogue précédente. Voici un exemple du contenu d'un fichier condensé. L'ordre des champs peut varier en fonction de votre navigateur.

```
{
  "digest": "42zaJOfV8iGutVGNaIuzQWhD5Xb/5B9lScHnvxPXm9E=",
  "digestTipAddress": "{strandId:\"BlFTjlSXze9BIh1KOszcE3\",sequenceNo:73}",
  "ledger": "vehicle-registration",
  "date": "2019-04-17T16:57:26.749Z"
```

```
}
```

6. Enregistrez ce fichier pour pouvoir y accéder ultérieurement. Au cours des étapes suivantes, vous utiliserez ce fichier pour vérifier la révision d'un document.

Après avoir enregistré un résumé du registre, vous pouvez démarrer le processus de vérification de la révision d'un document par rapport à ce condensé.

> **Note**
>
> Dans un cas d'utilisation en production à des fins de vérification, vous utilisez un condensé précédemment enregistré plutôt que d'effectuer les deux tâches consécutivement. Il est recommandé de demander et d'enregistrer le résumé dès qu'une révision que vous souhaitez vérifier ultérieurement est écrite dans le journal.

Pour vérifier la révision d'un document

1. Tout d'abord, recherchez dans votre registre`id` la`blockAddress` fin de la révision du document que vous souhaitez vérifier. Ces champs sont inclus dans les métadonnées du document, que vous pouvez consulter dans la vue validée.

   Le document`id` est une chaîne d'identification unique attribuée par le système. `blockAddress`Il s'agit d'une structure ionique qui spécifie l'emplacement du bloc où la révision a été validée.

   Dans le panneau de navigation de la console QLDB, choisissez l'éditeur PartiQL.
2. Choisissez le`vehicle-registration` registre.
3. Dans la fenêtre de l'éditeur de requêtes, entrez l'instruction suivante, puis choisissez Exécuter.

   ```
   SELECT r.metadata.id, r.blockAddress
   FROM _ql_committed_VehicleRegistration AS r
   WHERE r.data.VIN = '1N4AL11D75C109151'
   ```
4. Copiez et enregistrez les`blockAddress` valeurs`id` et renvoyées par votre requête. Veillez à omettre les guillemets doubles pour le`id` champ. Dans Amazon Ion, les types de données de chaîne sont délimités par des guillemets doubles.
5. Maintenant qu'une révision de document est sélectionnée, vous pouvez commencer le processus de vérification.

Dans le panneau de navigation, choisissez Vérification.

6. Dans le formulaire Vérifier le document, sous Spécifiez le document que vous souhaitez vérifier, entrez les paramètres de saisie suivants :

   - Ledger — Choisissez`vehicle-registration`.
   - Adresse de bloc :`blockAddress` valeur renvoyée par votre requête à l'étape 3.
   - ID du document :`id` valeur renvoyée par votre requête à l'étape 3.

7. Sous Spécifier le condensé à utiliser pour la vérification, sélectionnez le condensé que vous avez précédemment enregistré en choisissant Choisir un condensé. Si le fichier est valide, tous les champs de résumé de votre console sont automatiquement renseignés. Vous pouvez également copier et coller manuellement les valeurs suivantes directement à partir de votre fichier de résumé :

   - Résumé :`digest` valeur de votre fichier de résumé.
   - Adresse du résumé :`digestTipAddress` valeur de votre fichier de résumé.

8. Passez en revue les paramètres d'entrée du document et du condensé, puis choisissez Vérifier.

   La console automatise deux étapes pour vous :

   a. Demandez une preuve à QLDB pour le document que vous avez spécifié.

   b. Utilisez la preuve renvoyée par QLDB pour appeler une API côté client, qui vérifie la révision de votre document par rapport au condensé fourni.

   La console affiche les résultats de votre demande dans la fiche des résultats de vérification. Pour plus d'informations, veuillez consulter Résultats de vérification.

9. Pour tester la logique de vérification, répétez les étapes 6 à 8 de la section Pour vérifier la révision d'un document, mais modifiez un seul caractère dans la chaîne d'entrée du Digest. Cela devrait entraîner l'échec de votre demande de vérification avec un message d'erreur approprié.

Si vous n'avez plus besoin d'utiliser le`vehicle-registration` registre, passez àÉtape 7 (optionnelle) : Nettoyer les ressources.

Étape 7 (optionnelle) : Nettoyer les ressources

Vous pouvez continuer à utiliser le`vehicle-registration` registre. Toutefois, si vous n'en avez plus besoin, vous devez le supprimer.

Si la protection contre la suppression est activée pour votre registre, vous devez commencer par la désactiver avant de pouvoir supprimer le registre à l'aide de l'API QLDB, QLDB, QLDB.AWS Command Line InterfaceAWS CLI

Pour supprimer le registre

1. Ouvrez la console Amazon QLDB à l'adresse https://console.aws.amazon.com/qldb.
2. Dans le panneau de navigation, choisissez Ledgers.
3. Si la protection contre la suppression est activée pour ce registre, vous devez commencer par la désactiver.

   Dans la liste des livres, sélectionnez`vehicle-registration`, puis choisissez Modifier le livre.
4. Sur la page Modifier le registre, désactivez la protection contre la suppression, puis choisissez Confirmer les modifications.
5. Dans la liste des livres, sélectionnez`vehicle-registration` à nouveau, puis choisissez Supprimer.
6. Confirmez cela**`delete vehicle-registration`** en saisissant le champ prévu à cet effet.

Pour en savoir plus sur l'utilisation des registres dans QLDB, consultezMise en route avec Amazon QLDB : étapes suivantes.

Mise en route avec Amazon QLDB : étapes suivantes

Pour de plus d'informations sur l'utilisation d'Amazon QLDB, consultez les rubriques suivantes :

- Utilisation des données et de l'historique dans Amazon QLDB
- Démarrage QLDB
- Modèle de mise QLDB simultansimultansimultansimultansimultansimultansimultansimultanéité
- Exportation de données de journal depuis Amazon QLDB
- Diffusion en continu de données de journaux depuis Amazon QLDB

- Vérification des données dans Amazon QLDB
- Référence Amazon QLDB

Démarrage QLDB

Ce chapitre contient des didacticiels pratiques qui vous aideront à développer avec Amazon QLDB. Le pilote est construit sur leAWS SDK, qui prend en charge l'interaction avec l'API QLDB.

Abstraction de session QLDB

Le pilote fournit une couche d'abstraction de haut niveau au-dessus de l'API de données transactionnelles (session QLDB). Il rationalise le processus d'exécution des instructions partiQL sur les données du registre en gérant les appels SendCommandd'API. Ces appels d'API nécessitent plusieurs paramètres que le pilote gère à votre place, notamment la gestion des sessions, des transactions et la politique de nouvelle tentative en cas d'erreur. Le pilote dispose également d'optimisations de performances et applique les meilleures pratiques pour interagir avec QLDB.

> **Note**
>
> Pour interagir avec les opérations de l'API de gestion des ressources répertoriées dans la référence de l'API Amazon QLDB, vous utilisez directement leAWS SDK au lieu du pilote. Vous utilisez l'API de gestion uniquement pour gérer les ressources du registre et pour les opérations de données non transactionnelles, telles que l'exportation, le streaming et la vérification des données.

Assistance Amazon Ion

En outre, le pilote utilise les bibliothèques Amazon Ion pour prendre en charge la gestion des données Ion lors de l'exécution de transactions. Ces bibliothèques se chargent également de calculer le hachage des valeurs d'ions. QLDB a besoin de ces hachages ioniques pour vérifier l'intégrité des demandes de transactions de données.

Terminologie des conducteurs

Cet outil est appelé pilote car il est comparable à d'autres pilotes de base de données qui fournissent des interfaces conviviales pour les développeurs. Ces pilotes encapsulent de la même manière une logique qui convertit un ensemble standard de commandes et de fonctions en appels spécifiques requis par l'API de bas niveau du service.

Le pilote est open source GitHub et est disponible pour les langages de programmation suivants :

- pilote

- pilote .NET Driver Driver Lo
- Go Driver
- pilote Node.js
- pilote Python

Pour obtenir des informations générales sur les pilotes de tous les langages de programmation pris en charge, ainsi que des didacticiels supplémentaires, consultez les rubriques suivantes :

- Gestion des sessions avec le chauffeur
- Recommandations du conducteur
- Stratégie de nouvelle tentative de tentative de pilote
- Erreurs courantes
- Démarche d'un exemple d'application
- Utilisation d'Amazon Ion
- Obtention des statistiques d'instruction PartiQL

Pilote Amazon QLDB pour Java

Pour utiliser les données de votre registre, vous pouvez vous connecter à Amazon QLDB depuis votre application Java à l'aide d'un piloteAWS fourni. Les rubriques suivantes décrivent comment utiliser le pilote QLDB pour Java.

Rubriques

- Ressources pour les conducteurs
- Prérequis
- Configuration de vosAWS informations d'identification et de votre région par défaut
- Installation
- Pilote Amazon QLDB pour Java — Tutoriel de démarrage rapide
- Pilote Amazon QLDB pour Java — Référence du livre de recettes

Ressources pour les conducteurs

Pour plus d'informations sur les fonctionnalités prises en charge par le pilote Java

- Référence de l'API : 2.x, 1.x
- Code source du pilote (GitHub)
- Exemple de code source d'application (GitHub)
- Framework Ledger Loader (GitHub)
- Exemples de code Amazon Ion

Prérequis

Avant de commencer avec le pilote QLDB pour Java, vous devez effectuer les opérations suivantes :

1. Suivez les instructions deAWS configuration dansAccès à Amazon QLDB. Cela inclut les éléments suivants :
    1. S'inscrireAWS à
    2. Créer un utilisateur avec les autorisations QLDB.
    3. Accorder un accès par programmation
2. Configurez un environnement de développement Java en téléchargeant et en installant les éléments suivants :
    1. Kit de développement Java SE 8, tel qu'Amazon Corretto 8.
    2. (Facultatif) Environnement de développement intégré (IDE) Java de votre choix, tel qu'Eclipse ou IntelliJ.
3. Configurez votre environnement de développement pourAWS SDK for Java leConfiguration de vosAWS informations d'identification et de votre région par défaut.

Vous pouvez ensuite télécharger l'exemple d'application complet du didacticiel, ou vous pouvez uniquement installer le pilote dans un projet Java et exécuter des exemples de code courts.

- Pour installer le pilote QLDB et leAWS SDK for Java dans un projet existant, passez àInstallation.
- Pour configurer un projet et exécuter des exemples de code abrégé illustrant les transactions de données de base sur un registre, consultez leDacticiel de démarrage rapide.
- Pour exécuter des exemples plus détaillés d'opérations sur les données et les API de gestion dans l'exemple d'application complet du didacticiel, consultez leTutoriel Java.

Configuration de vosAWS informations d'identification et de votre région par défaut

Le pilote QLDB et le kit AWS SDK for JavaAWS Les exemples de code dans ce guide partent du principe que vous utilisez un fichierAWS d'informations d'identification, comme décrit dans la section Configuration des informations d'identification du Guide duAWS SDK for Java 2.x développeur.

Dans le cadre de ces étapes, vous devez également définir votre pointRégion AWS de terminaison QLDB par défaut. Les exemples de code se connectent à QLDB par défautRégion AWS. Pour obtenir la liste complète des régions dans lesquelles QLDB est disponible, consultez les points de terminaison et quotas Amazon QLDB dans le Références générales AWS.

Voici un exemple de fichier d'informations d'identification AWS nommé `~/.aws/credentials`, où le caractère tilde (~) représente votre répertoire de base.

```
[default]
aws_access_key_id = your_access_key_id
aws_secret_access_key = your_secret_access_key
```

Remplacez vos propres valeursAWS d'identification par les valeurs *`your_access_key_id`* et *`your_secret_access_key`*.

Installation

QLDB prend en charge les versions de pilotes Java suivantes et leurs dépendances en matière deAWS SDK.

Versions du pilote	AWS SDK	État	Dernière date de parution
1.x	AWS SDK for Java1.x	Version de production	20 mars 2020
2.x	AWS SDK for Java 2.x	Version de production	4 juin 2021

Pour installer le pilote QLDB, nous vous recommandons d'utiliser un système de gestion des dépendances, tel que Gradle ou Maven. Par exemple, ajoutez l'artefact suivant en tant que dépendance dans votre projet Java.

2.x

Gradle

Ajoutez cette dépendance dans votre fichier`build.gradle` de configuration.

```
dependencies {
    compile group: 'software.amazon.qldb', name: 'amazon-qldb-driver-java', version:
 '2.3.1'
}
```

Maven

Ajoutez cette dépendance dans votre fichier`pom.xml` de configuration.

```
<dependencies>
  <dependency>
    <groupId>software.amazon.qldb</groupId>
    <artifactId>amazon-qldb-driver-java</artifactId>
    <version>2.3.1</version>
  </dependency>
</dependencies>
```

Cet artefact inclut automatiquement le moduleAWS SDK for Java 2.x principal, les bibliothèques Amazon Ion et les autres dépendances requises.

1.x

Gradle

Ajoutez cette dépendance dans votre fichier`build.gradle` de configuration.

```
dependencies {
    compile group: 'software.amazon.qldb', name: 'amazon-qldb-driver-java', version:
 '1.1.0'
}
```

Maven

Ajoutez cette dépendance dans votre fichier`pom.xml` de configuration.

```
<dependencies>
  <dependency>
```

```
    <groupId>software.amazon.qldb</groupId>
    <artifactId>amazon-qldb-driver-java</artifactId>
    <version>1.1.0</version>
  </dependency>
</dependencies>
```

Cet artefact inclut automatiquement le moduleAWS SDK for Java principal, les bibliothèques Amazon Ion et les autres dépendances requises.

> **Important**
>
> Espace de noms Amazon Ion : lorsque vous importez des classes Amazon Ion dans votre application, vous devez utiliser le package qui se trouve sous l'espace de nomscom.amazon.ion. AWS SDK for JavaCela dépend d'un autre package Ion dans l'espace de nomssoftware.amazon.ion, mais il s'agit d'un package existant qui n'est pas compatible avec le pilote QLDB.

Pour des exemples de code courts expliquant comment exécuter des transactions de données de base sur un registre, consultez leRéférence du livre de recettes.

Autres bibliothèques

Le cas échéant, vous pouvez également ajouter les bibliothèques utiles suivantes. Ces artefacts sont des dépendances requises dans l'Tutoriel Javaexemple d'application.

1. aws-java-sdk-qldb— Le module QLDB duAWS SDK for Java. La version minimale prise en charge par QLDB est1.11.785.

   Utilisez ce module dans votre application pour interagir directement avec les opérations de l'API de gestion répertoriées dans leRéférence d'API Amazon QLDB.
2. jackson-dataformat-ion— Module de format de données Jackson de FasterXML pour Ion. L'exemple d'application nécessite une version2.10.0 ou une version ultérieure.

Gradle

Ajoutez ces dépendances dans votre fichierbuild.gradle de configuration.

```
dependencies {
```

```
    compile group: 'com.amazonaws', name: 'aws-java-sdk-qldb', version: '1.11.785'
    compile group: 'com.fasterxml.jackson.dataformat', name: 'jackson-dataformat-
ion', version: '2.10.0'
}
```

Maven

Ajoutez ces dépendances dans votre fichier`pom.xml` de configuration.

```
<dependencies>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-java-sdk-qldb</artifactId>
    <version>1.11.785</version>
  </dependency>
  <dependency>
    <groupId>com.fasterxml.jackson.dataformat</groupId>
    <artifactId>jackson-dataformat-ion</artifactId>
    <version>2.10.0</version>
  </dependency>
</dependencies>
```

Pilote Amazon QLDB pour Java — Tutoriel de démarrage rapide

Dans ce tutoriel, vous apprendrez comment configurer une application simple à l'aide de la dernière version du pilote Amazon QLDB pour Java. Ce guide inclut les étapes d'installation du pilote et des exemples de base de création, mise à jour et suppression (CRUD). Pour des exemples plus détaillés illustrant ces opérations dans un exemple d'application complet, consultez leTutoriel Java.

Rubriques

- Exécution de l'application complète

Prérequis

Avant de commencer, veillez à exécuter les actions suivantes :

1. Complétez lePrérequis pour le pilote Java, si vous ne l'avez pas déjà fait. Cela inclut l'inscriptionAWS, l'octroi d'un accès programmatique pour le développement et l'installation d'un environnement de développement intégré (IDE) Java.
2. Créez un registre nomméquick-start.

   Pour savoir comment créer un registre, consultezOpérations de base pour les registres Amazon QLDB ouÉtape 1 : Créer un nouveau registre dans Prise en main de la console.

Étape 1 : Configurer votre projet

Tout d'abord, configurez votre projet Java. Nous vous recommandons d'utiliser le système de gestion des dépendances Maven pour ce didacticiel.

> **Note**
>
> Si vous utilisez un IDE doté de fonctionnalités permettant d'automatiser ces étapes de configuration, vous pouvez passer directement àÉtape 2 : Initialiser le pilote.

1. Créez un dossier pour votre application.

   ```
   $ mkdir myproject
   $ cd myproject
   ```

2. Saisissez la commande suivante pour initialiser votre projet à partir d'un modèle Maven. Remplacez le *package du projet, le nom du* projet et le *modèle maven* par vos propres valeurs, le cas échéant.

   ```
   $ mvn archetype:generate
     -DgroupId=project-package \
     -DartifactId=project-name \
     -DarchetypeArtifactId=maven-template \
   ```

```
  -DinteractiveMode=false
```

Pour *maven-template*, vous pouvez utiliser le modèle Maven de base :`maven-archetype-quickstart`

3. Pour ajouter le pilote QLDB pour Java en tant que dépendance de projet, accédez au`pom.xml` fichier que vous venez de créer et ajoutez l'artefact suivant.

```
<dependency>
    <groupId>software.amazon.qldb</groupId>
    <artifactId>amazon-qldb-driver-java</artifactId>
    <version>2.3.1</version>
</dependency>
```

Cet artefact inclut automatiquement le module AWS SDK for Java 2.xprincipal, les bibliothèques Amazon Ion et les autres dépendances requises. Votre`pom.xml` fichier doit maintenant se présenter comme suit :

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/
maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>software.amazon.qldb</groupId>
  <artifactId>qldb-quickstart</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>qldb-quickstart</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>software.amazon.qldb</groupId>
        <artifactId>amazon-qldb-driver-java</artifactId>
        <version>2.3.1</version>
    </dependency>
  </dependencies>
```

```
</project>
```

4. Ouvrez le fichier `App.java`.

   Ajoutez ensuite progressivement les exemples de code dans les étapes suivantes pour essayer certaines opérations CRUD de base. Vous pouvez également ignorer le step-by-step didacticiel et exécuter l'application complète.

Étape 2 : Initialiser le pilote

Initialisez une instance du pilote qui se connecte au registre nomméquick-start. Ajoutez le code suivant à votreApp.java fichier.

```
import java.util.*;
import com.amazon.ion.*;
import com.amazon.ion.system.*;
import software.amazon.awssdk.services.qldbsession.QldbSessionClient;
import software.amazon.qldb.*;

public final class App {
    public static IonSystem ionSys = IonSystemBuilder.standard().build();
    public static QldbDriver qldbDriver;

    public static void main(final String... args) {
        System.out.println("Initializing the driver");
        qldbDriver = QldbDriver.builder()
            .ledger("quick-start")
            .transactionRetryPolicy(RetryPolicy
                .builder()
                .maxRetries(3)
                .build())
            .sessionClientBuilder(QldbSessionClient.builder())
            .build();
    }
}
```

Étape 3 : création d'une table et d'un index

L'exemple de code suivant montre comment exécuterCREATE TABLE lesCREATE INDEX instructions and.

Dans la`main` méthode, ajoutez le code suivant qui crée une table nommée`People` et un index pour le`lastName` champ de cette table. Les index sont nécessaires pour optimiser les performances des requêtes et contribuer à limiter les exceptions de conflit liées au contrôle optimiste de la concurrence (OCC).

```
// Create a table and an index in the same transaction
qldbDriver.execute(txn -> {
    System.out.println("Creating a table and an index");
    txn.execute("CREATE TABLE People");
    txn.execute("CREATE INDEX ON People(lastName)");
});
```

Étape 4 : Insertion d'un document

L'exemple de code suivant montre comment exécuter une`INSERT` instruction. QLDB prend en charge le langage de requête PartiQL (compatible SQL) et le format de données Amazon Ion (surensemble de JSON).

Ajoutez le code suivant pour insérer un document dans le`People` tableau.

```
// Insert a document
qldbDriver.execute(txn -> {
    System.out.println("Inserting a document");
    IonStruct person = ionSys.newEmptyStruct();
    person.put("firstName").newString("John");
    person.put("lastName").newString("Doe");
    person.put("age").newInt(32);
    txn.execute("INSERT INTO People ?", person);
});
```

Cet exemple utilise un point d'interrogation (?) comme espace réservé variable pour transmettre les informations du document à l'instruction. Lorsque vous utilisez des espaces réservés, vous devez transmettre une valeur de type`IonValue`.

> **Tip**
>
> Pour insérer plusieurs documents à l'aide d'une seuleINSERT instruction, vous pouvez transmettre un paramètre de type IonList(explicitement converti en tant que`IonValue`) à l'instruction comme suit.
>
> ```
> // people is an IonList explicitly cast as an IonValue
> ```

```
txn.execute("INSERT INTO People ?", (IonValue) people);
```

Vous ne placez pas l'espace réservé à la variable (?) entre crochets (`<<...>>`) lorsque vous transmettez un`IonList`. Dans les instructions manuelles de PartiQL, les crochets doubles indiquent une collection non ordonnée appelée sac.

Étape 5 : interroger le document

L'exemple de code suivant montre comment exécuter une`SELECT` instruction.

Ajoutez le code suivant qui interroge un document à partir du`People` tableau.

```
// Query the document
qldbDriver.execute(txn -> {
    System.out.println("Querying the table");
    Result result = txn.execute("SELECT * FROM People WHERE lastName = ?",
            ionSys.newString("Doe"));
    IonStruct person = (IonStruct) result.iterator().next();
    System.out.println(person.get("firstName")); // prints John
    System.out.println(person.get("lastName")); // prints Doe
    System.out.println(person.get("age")); // prints 32
});
```

Étape 6 : Mettre à jour le document

L'exemple de code suivant montre comment exécuter une`UPDATE` instruction.

1. Ajoutez le code suivant pour mettre à jour un document dans le`People` tableau en le mettant à jour`age` vers`42`.

   ```
   // Update the document
   qldbDriver.execute(txn -> {
       System.out.println("Updating the document");
       final List<IonValue> parameters = new ArrayList<>();
       parameters.add(ionSys.newInt(42));
       parameters.add(ionSys.newString("Doe"));
       txn.execute("UPDATE People SET age = ? WHERE lastName = ?", parameters);
   });
   ```

2. Interrogez à nouveau le document pour voir la valeur mise à jour.

```
// Query the updated document
qldbDriver.execute(txn -> {
    System.out.println("Querying the table for the updated document");
    Result result = txn.execute("SELECT * FROM People WHERE lastName = ?",
            ionSys.newString("Doe"));
    IonStruct person = (IonStruct) result.iterator().next();
    System.out.println(person.get("firstName")); // prints John
    System.out.println(person.get("lastName")); // prints Doe
    System.out.println(person.get("age")); // prints 32
});
```

3. Utilisez Maven ou votre IDE pour compiler et exécuter leApp.java fichier.

Exécution de l'application complète

L'exemple de code suivant représente la version complète de l'App.javaapplication. Au lieu de suivre les étapes précédentes individuellement, vous pouvez également copier et exécuter cet exemple de code du début à la fin. Cette application montre certaines opérations CRUD de base sur le registre nomméquick-start.

> **Note**
>
> Avant d'exécuter ce code, assurez-vous qu'aucune table active n'est déjà nomméePeople dans lequick-start registre.
> Sur la première ligne, remplacez *project-package* par lagroupId valeur que vous avez utilisée pour la commande Maven dansÉtape 1 : Configurer votre projet .

```
package project-package;

import java.util.*;
import com.amazon.ion.*;
import com.amazon.ion.system.*;
import software.amazon.awssdk.services.qldbsession.QldbSessionClient;
import software.amazon.qldb.*;

public class App {
    public static IonSystem ionSys = IonSystemBuilder.standard().build();
    public static QldbDriver qldbDriver;
```

```
    public static void main(final String... args) {
        System.out.println("Initializing the driver");
        qldbDriver = QldbDriver.builder()
            .ledger("quick-start")
            .transactionRetryPolicy(RetryPolicy
                .builder()
                .maxRetries(3)
                .build())
            .sessionClientBuilder(QldbSessionClient.builder())
            .build();

        // Create a table and an index in the same transaction
        qldbDriver.execute(txn -> {
            System.out.println("Creating a table and an index");
            txn.execute("CREATE TABLE People");
            txn.execute("CREATE INDEX ON People(lastName)");
        });

        // Insert a document
        qldbDriver.execute(txn -> {
            System.out.println("Inserting a document");
            IonStruct person = ionSys.newEmptyStruct();
            person.put("firstName").newString("John");
            person.put("lastName").newString("Doe");
            person.put("age").newInt(32);
            txn.execute("INSERT INTO People ?", person);
        });

        // Query the document
        qldbDriver.execute(txn -> {
            System.out.println("Querying the table");
            Result result = txn.execute("SELECT * FROM People WHERE lastName = ?",
                    ionSys.newString("Doe"));
            IonStruct person = (IonStruct) result.iterator().next();
            System.out.println(person.get("firstName")); // prints John
            System.out.println(person.get("lastName")); // prints Doe
            System.out.println(person.get("age")); // prints 32
        });

        // Update the document
        qldbDriver.execute(txn -> {
            System.out.println("Updating the document");
            final List<IonValue> parameters = new ArrayList<>();
            parameters.add(ionSys.newInt(42));
```

```
            parameters.add(ionSys.newString("Doe"));
            txn.execute("UPDATE People SET age = ? WHERE lastName = ?", parameters);
        });

        // Query the updated document
        qldbDriver.execute(txn -> {
            System.out.println("Querying the table for the updated document");
            Result result = txn.execute("SELECT * FROM People WHERE lastName = ?",
                    ionSys.newString("Doe"));
            IonStruct person = (IonStruct) result.iterator().next();
            System.out.println(person.get("firstName")); // prints John
            System.out.println(person.get("lastName")); // prints Doe
            System.out.println(person.get("age")); // prints 42
        });
    }
}
```

Utilisez Maven ou votre IDE pour compiler et exécuter le`App.java` fichier.

Pilote Amazon QLDB pour Java — Référence du livre de recettes

Ce guide de référence présente des cas d'utilisation courants du pilote Amazon QLDB pour Java. Il fournit des exemples de code Java montrant comment utiliser le pilote pour exécuter des opérations de base. Il inclut également des exemples de code pour le traitement des données Amazon Ion. En outre, ce guide met en évidence les meilleures pratiques pour rendre les transactions idempotentes et mettre en œuvre des contraintes d'unicité.

> **Note**
>
> Le cas échéant, certains cas d'utilisation comportent des exemples de code différents pour chaque version majeure prise en charge du pilote QLDB pour Java.

Table des matières

Importation du pilote

L'exemple de code suivant importe le pilote, le client de session QLDB, les packages Amazon Ion et d'autres dépendances connexes.

2.x

```
import com.amazon.ion.IonStruct;
import com.amazon.ion.IonSystem;
import com.amazon.ion.IonValue;
import com.amazon.ion.system.IonSystemBuilder;

import software.amazon.awssdk.services.qldbsession.QldbSessionClient;
import software.amazon.qldb.QldbDriver;
import software.amazon.qldb.Result;
```

1.x

```
import com.amazon.ion.IonStruct;
import com.amazon.ion.IonSystem;
import com.amazon.ion.IonValue;
import com.amazon.ion.system.IonSystemBuilder;
```

```
import com.amazonaws.services.qldbsession.AmazonQLDBSessionClientBuilder;
import software.amazon.qldb.PooledQldbDriver;
import software.amazon.qldb.Result;
```

Instanciation du pilote

L'exemple de code suivant crée une instance de pilote qui se connecte à un nom de registre spécifié et utilise une logique de nouvelle tentative spécifiée avec une limite de tentatives personnalisée.

> **Note**
>
> Cet exemple instancie également un objet système Amazon Ion (`IonSystem`). Vous avez besoin de cet objet pour traiter les données Ion lors de l'exécution de certaines opérations de données dans cette référence. Pour en savoir plus, consultez Utilisation d'Amazon Ion.

2.x

```
QldbDriver qldbDriver = QldbDriver.builder()
    .ledger("vehicle-registration")
    .transactionRetryPolicy(RetryPolicy
        .builder()
        .maxRetries(3)
        .build())
    .sessionClientBuilder(QldbSessionClient.builder())
    .build();

IonSystem SYSTEM = IonSystemBuilder.standard().build();
```

1.x

```
PooledQldbDriver qldbDriver = PooledQldbDriver.builder()
    .withLedger("vehicle-registration")
    .withRetryLimit(3)
    .withSessionClientBuilder(AmazonQLDBSessionClientBuilder.standard())
    .build();

IonSystem SYSTEM = IonSystemBuilder.standard().build();
```

Opérations CRUD

QLDB exécute des opérations de création, lecture, mise à jour et suppression.

> **Warning**
>
> À titre de bonne pratique, faites de vos transactions d'écriture.

Rendre les transactions idempotentes

Nous vous recommandons de rendre les transactions d'écriture idempotentes afin d'éviter tout effet secondaire inattendu en cas de nouvelles tentatives. Une transaction est idempotente si elle peut être exécutée plusieurs fois et produire des résultats identiques à chaque fois.

Prenons l'exemple d'une transaction qui insère un document dans une table nomméePerson. La transaction doit d'abord vérifier si le document existe déjà dans le tableau. Sans cette vérification, le tableau risque de contenir des documents dupliqués.

Supposons que QLDB valide avec succès la transaction côté serveur, mais que le client expire en attendant une réponse. Si la transaction n'est pas idempotente, le même document peut être inséré plusieurs fois en cas de nouvelle tentative.

Utilisation d'index pour éviter l'analyse complète des tables

Nous vous recommandons également d'exécuter des instructions avec une clause deWHERE prédicat à l'aide d'un opérateur d'égalité sur un champ indexé ou un identifiant de document, par exemple,`WHERE indexedField = 123` ou`WHERE indexedField IN (456, 789)`. Sans cette recherche indexée, QLDB doit effectuer une analyse des tables, ce qui peut entraîner des délais d'expiration des transactions ou des conflits de contrôle optimiste de la concurrence (OCC).

Pour plus d'informations sur l'OCC, consultezModèle de mise QLDB simultansimultansimultansimultansimultansimultansimultansimultansimultanéité.

Transactions créées implicitement

La méthode QldbDriver.execute accepte une fonction lambda qui reçoit une instance d'Executor, que vous pouvez utiliser pour exécuter des instructions. L'`Executor`instance encapsule une transaction créée implicitement.

Vous pouvez exécuter des instructions dans la fonction Lambda à l'aide de la`Executor.execute` méthode. Le pilote valide implicitement la transaction lorsque la fonction Lambda est renvoyée.

Les sections suivantes montrent comment exécuter des opérations CRUD de base, spécifier une logique de nouvelle tentative personnalisée et implémenter des contraintes d'unicité.

> **Note**
>
> Le cas échéant, ces sections fournissent des exemples de code de traitement de données Amazon Ion à l'aide de la bibliothèque Ion intégrée et de la bibliothèque de mappage Jackson Ion. Pour en savoir plus, consultez Utilisation d'Amazon Ion.

Table des matières

Création de tables

```
qldbDriver.execute(txn -> {
    txn.execute("CREATE TABLE Person");
});
```

Créer des index

```
qldbDriver.execute(txn -> {
    txn.execute("CREATE INDEX ON Person(GovId)");
```

```
});
```

Lire des documents

```
// Assumes that Person table has documents as follows:
// { GovId: "TOYENC486FH", FirstName: "Brent" }

qldbDriver.execute(txn -> {
    Result result = txn.execute("SELECT * FROM Person WHERE GovId = 'TOYENC486FH'");
    IonStruct person = (IonStruct) result.iterator().next();
    System.out.println(person.get("GovId")); // prints TOYENC486FH
    System.out.println(person.get("FirstName")); // prints Brent
});
```

Utilisation des paramètres de requête

L'exemple de code suivant utilise un paramètre de requête de type Ion.

```
qldbDriver.execute(txn -> {
    Result result = txn.execute("SELECT * FROM Person WHERE GovId = ?",
            SYSTEM.newString("TOYENC486FH"));
    IonStruct person = (IonStruct) result.iterator().next();
    System.out.println(person.get("GovId")); // prints TOYENC486FH
    System.out.println(person.get("FirstName")); // prints Brent
});
```

L'exemple de code suivant utilise plusieurs paramètres de requête.

```
qldbDriver.execute(txn -> {
    Result result = txn.execute("SELECT * FROM Person WHERE GovId = ? AND FirstName
 = ?",
            SYSTEM.newString("TOYENC486FH"),
            SYSTEM.newString("Brent"));
    IonStruct person = (IonStruct) result.iterator().next();
    System.out.println(person.get("GovId")); // prints TOYENC486FH
    System.out.println(person.get("FirstName")); // prints Brent
});
```

L'exemple de code suivant utilise une liste de paramètres de requête.

```
qldbDriver.execute(txn -> {
    final List<IonValue> parameters = new ArrayList<>();
```

```
    parameters.add(SYSTEM.newString("TOYENC486FH"));
    parameters.add(SYSTEM.newString("ROEE1"));
    parameters.add(SYSTEM.newString("YH844"));
    Result result = txn.execute("SELECT * FROM Person WHERE GovId IN (?,?,?)",
 parameters);
    IonStruct person = (IonStruct) result.iterator().next();
    System.out.println(person.get("GovId")); // prints TOYENC486FH
    System.out.println(person.get("FirstName")); // prints Brent
});
```

Utilisation du mappeur Jackson

```
// Assumes that Person table has documents as follows:
// {GovId: "TOYENC486FH", FirstName: "Brent" }

qldbDriver.execute(txn -> {
    try {
        Result result = txn.execute("SELECT * FROM Person WHERE GovId =
 'TOYENC486FH'");
        Person person = MAPPER.readValue(result.iterator().next(), Person.class);
        System.out.println(person.getFirstName()); // prints Brent
        System.out.println(person.getGovId()); // prints TOYENC486FH
    } catch (IOException e) {
        e.printStackTrace();
    }
});
```

Utilisation des paramètres de requête

L'exemple de code suivant utilise un paramètre de requête de type Ion.

```
qldbDriver.execute(txn -> {
    try {
        Result result = txn.execute("SELECT * FROM Person WHERE GovId = ?",
                MAPPER.writeValueAsIonValue("TOYENC486FH"));
        Person person = MAPPER.readValue(result.iterator().next(), Person.class);
        System.out.println(person.getFirstName()); // prints Brent
        System.out.println(person.getGovId()); // prints TOYENC486FH
    } catch (IOException e) {
        e.printStackTrace();
    }
});
```

L'exemple de code suivant utilise plusieurs paramètres de requête.

```
qldbDriver.execute(txn -> {
    try {
        Result result = txn.execute("SELECT * FROM Person WHERE GovId = ? AND FirstName
 = ?",
                MAPPER.writeValueAsIonValue("TOYENC486FH"),
                MAPPER.writeValueAsIonValue("Brent"));
        Person person = MAPPER.readValue(result.iterator().next(), Person.class);
        System.out.println(person.getFirstName()); // prints Brent
        System.out.println(person.getGovId()); // prints TOYENC486FH
    } catch (IOException e) {
        e.printStackTrace();
    }
});
```

L'exemple de code suivant utilise une liste de paramètres de requête.

```
qldbDriver.execute(txn -> {
    try {
        final List<IonValue> parameters = new ArrayList<>();
        parameters.add(MAPPER.writeValueAsIonValue("TOYENC486FH"));
        parameters.add(MAPPER.writeValueAsIonValue("ROEE1"));
        parameters.add(MAPPER.writeValueAsIonValue("YH844"));
        Result result = txn.execute("SELECT * FROM Person WHERE GovId IN (?,?,?)",
 parameters);
        Person person = MAPPER.readValue(result.iterator().next(), Person.class);
        System.out.println(person.getFirstName()); // prints Brent
        System.out.println(person.getGovId()); // prints TOYENC486FH
    } catch (IOException e) {
        e.printStackTrace();
    }
});
```

> **Note**
>
> Lorsque vous exécutez une requête sans recherche indexée, elle appelle une analyse complète de la table. Dans cet exemple, nous recommandons de disposer d'un index sur le`GovId` terrain afin d'optimiser les performances. Si aucun index n'est activé`GovId`, les requêtes peuvent présenter une latence plus importante et peuvent également entraîner des exceptions de conflit OCC ou des délais de transaction.

Insertion de documents

L'exemple de code suivant insère des types de données Ion.

```
qldbDriver.execute(txn -> {
    // Check if a document with GovId:TOYENC486FH exists
    // This is critical to make this transaction idempotent
    Result result = txn.execute("SELECT * FROM Person WHERE GovId = ?",
            SYSTEM.newString("TOYENC486FH"));
    // Check if there is a result
    if (!result.iterator().hasNext()) {
        IonStruct person = SYSTEM.newEmptyStruct();
        person.put("GovId").newString("TOYENC486FH");
        person.put("FirstName").newString("Brent");
        // Insert the document
        txn.execute("INSERT INTO Person ?", person);
    }
});
```

Utilisation du mappeur Jackson

L'exemple de code suivant insère des types de données Ion.

```
qldbDriver.execute(txn -> {
    try {
        // Check if a document with GovId:TOYENC486FH exists
        // This is critical to make this transaction idempotent
        Result result = txn.execute("SELECT * FROM Person WHERE GovId = ?",
                MAPPER.writeValueAsIonValue("TOYENC486FH"));
        // Check if there is a result
        if (!result.iterator().hasNext()) {
            // Insert the document
            txn.execute("INSERT INTO Person ?",
                    MAPPER.writeValueAsIonValue(new Person("Brent", "TOYENC486FH")));
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
});
```

Cette transaction insère un document dans le`Person` tableau. Avant de l'insérer, il vérifie d'abord si le document existe déjà dans le tableau. Ce contrôle confère à la transaction un caractère

idempotent. Même si vous exécutez cette transaction plusieurs fois, elle n'aura aucun effet secondaire imprévu.

> **Note**
>
> Dans cet exemple, nous recommandons de disposer d'un index sur le`GovId` terrain afin d'optimiser les performances. Si aucun index n'est activé`GovId`, les instructions peuvent présenter une latence plus importante et peuvent également entraîner des exceptions de conflit OCC ou des délais de transaction.

Insertion de plusieurs documents dans une seule déclaration

Pour insérer plusieurs documents à l'aide d'une seuleINSERT instruction, vous pouvez transmettre un paramètre de type IonList(explicitement converti en tant que`IonValue`) à l'instruction comme suit.

```
// people is an IonList explicitly cast as an IonValue
txn.execute("INSERT INTO People ?", (IonValue) people);
```

Vous ne placez pas l'espace réservé à la variable (?) entre crochets (`<<...>>`) lorsque vous transmettez un`IonList`. Dans les instructions manuelles de PartiQL, les crochets doubles indiquent une collection non ordonnée appelée sac.

Pourquoi le casting explicite est-il requis ?

La méthode TransactionExecutor.execute est surchargée. Il accepte un nombre variable d'`IonValue`arguments (varargs) ou un seul`List<IonValue>` argument. Dans ion-java,`IonList` est implémenté en tant que`List<IonValue>`.

Java utilise par défaut l'implémentation de méthode la plus spécifique lorsque vous appelez une méthode surchargée. Dans ce cas, lorsque vous passez un`IonList` paramètre, il utilise par défaut la méthode qui prend un`List<IonValue>`. Lorsqu'elle est invoquée, cette implémentation de méthode transmet les`IonValue` éléments de la liste sous forme de valeurs distinctes. Ainsi, pour invoquer la méthode varargs à la place, vous devez explicitement convertir un`IonList` paramètre en tant que`IonValue`.

Mise à jour des documents

```
qldbDriver.execute(txn -> {
```

```
    final List<IonValue> parameters = new ArrayList<>();
    parameters.add(SYSTEM.newString("John"));
    parameters.add(SYSTEM.newString("TOYENC486FH"));
    txn.execute("UPDATE Person SET FirstName = ? WHERE GovId = ?", parameters);
});
```

Utilisation du mappeur Jackson

```
qldbDriver.execute(txn -> {
    try {
        final List<IonValue> parameters = new ArrayList<>();
        parameters.add(MAPPER.writeValueAsIonValue("John"));
        parameters.add(MAPPER.writeValueAsIonValue("TOYENC486FH"));
        txn.execute("UPDATE Person SET FirstName = ? WHERE GovId = ?", parameters);
    } catch (IOException e) {
        e.printStackTrace();
    }
});
```

> **Note**
>
> Dans cet exemple, nous recommandons de disposer d'un index sur le`GovId` terrain afin d'optimiser les performances. Si aucun index n'est activé`GovId`, les instructions peuvent présenter une latence plus importante et peuvent également entraîner des exceptions de conflit OCC ou des délais de transaction.

Supprimer des documents

```
qldbDriver.execute(txn -> {
    txn.execute("DELETE FROM Person WHERE GovId = ?",
            SYSTEM.newString("TOYENC486FH"));
});
```

Utilisation du mappeur Jackson

```
qldbDriver.execute(txn -> {
    try {
        txn.execute("DELETE FROM Person WHERE GovId = ?",
                MAPPER.writeValueAsIonValue("TOYENC486FH"));
```

```
    } catch (IOException e) {
        e.printStackTrace();
    }
});
```

> **Note**
>
> Dans cet exemple, nous recommandons de disposer d'un index sur le`GovId` terrain afin d'optimiser les performances. Si aucun index n'est activé`GovId`, les instructions peuvent présenter une latence plus importante et peuvent également entraîner des exceptions de conflit OCC ou des délais de transaction.

Exécution de plusieurs relevés dans une transaction

```
// This code snippet is intentionally trivial. In reality you wouldn't do this because
 you'd
// set your UPDATE to filter on vin and insured, and check if you updated something or
 not.
public static boolean InsureCar(QldbDriver qldbDriver, final String vin) {
    final IonSystem ionSystem = IonSystemBuilder.standard().build();
    final IonString ionVin = ionSystem.newString(vin);

    return qldbDriver.execute(txn -> {
        Result result = txn.execute(
                "SELECT insured FROM Vehicles WHERE vin = ? AND insured = FALSE",
                ionVin);
        if (!result.isEmpty()) {
            txn.execute("UPDATE Vehicles SET insured = TRUE WHERE vin = ?", ionVin);
            return true;
        }
        return false;
    });
}
```

Nouvelle.

La`execute` méthode du pilote possède un mécanisme de nouvelle tentative intégré qui réessaie la transaction si une exception pouvant être réessayée se produit (par exemple, des délais d'attente ou des conflits OCC).

2.x

Le nombre maximum de nouvelles tentatives et la stratégie de nouvelles.

La limite de tentatives par défaut est de4, et la stratégie d'annulation par défaut est DefaultQldbTransactionBackoffStrategy. Vous pouvez définir la configuration des nouvelles tentatives par instance de pilote et également par transaction en utilisant une instance de RetryPolicy.

L'exemple de code suivant spécifie une logique de nouvelle tentative avec une limite de tentatives personnalisée et une stratégie d'arrêt personnalisée pour une instance du pilote.

```
public void retry() {
    QldbDriver qldbDriver = QldbDriver.builder()
            .ledger("vehicle-registration")
            .transactionRetryPolicy(RetryPolicy.builder()
                    .maxRetries(2)
                    .backoffStrategy(new CustomBackOffStrategy()).build())
            .sessionClientBuilder(QldbSessionClient.builder())
            .build();
}

private class CustomBackOffStrategy implements BackoffStrategy {

    @Override
    public Duration calculateDelay(RetryPolicyContext retryPolicyContext) {
        return Duration.ofMillis(1000 * retryPolicyContext.retriesAttempted());
    }
}
```

L'exemple de code suivant spécifie une logique de nouvelle tentative avec une limite de tentatives personnalisée et une stratégie d'annulation personnalisée pour une transaction particulière. Cette configuration`execute` remplace la logique de nouvelle tentative définie pour l'instance du pilote.

```
public void retry() {
    Result result = qldbDriver.execute(txn -> { txn.execute("SELECT * FROM Person
 WHERE GovId = ?",
            SYSTEM.newString("TOYENC486FH")); },
            RetryPolicy.builder()
                    .maxRetries(2)
                    .backoffStrategy(new CustomBackOffStrategy())
                    .build());
```

```
}

private class CustomBackOffStrategy implements BackoffStrategy {

    // Configuring a custom backoff which increases delay by 1s for each attempt.
    @Override
    public Duration calculateDelay(RetryPolicyContext retryPolicyContext) {
        return Duration.ofMillis(1000 * retryPolicyContext.retriesAttempted());
    }
}
```

1.x

Le nombre maximum de nouvelles tentatives. Vous pouvez configurer la limite de tentatives en définissant la`retryLimit` propriété lors de l'initialisation`PooledQldbDriver`.

La limite de tentatives par défaut est de`4`.

Mise en œuvre des contraintes d'unicité

QLDB ne prend pas en charge les index uniques, mais vous pouvez implémenter ce comportement dans votre application.

Supposons que vous souhaitiez implémenter une contrainte d'unicité sur le`GovId` champ de la`Person` table. Pour ce faire, vous pouvez écrire une transaction qui effectue les actions suivantes :

1. Affirmez que la table ne contient aucun document existant portant une valeur spécifiée`GovId`.
2. Insérez le document si l'assertion est acceptée.

Si une transaction concurrente passe simultanément l'assertion, seule l'une des transactions sera validée avec succès. L'autre transaction échouera avec une exception de conflit OCC.

L'exemple de code suivant montre comment mettre en œuvre cette logique de contraintes d'unicité.

```
qldbDriver.execute(txn -> {
    Result result = txn.execute("SELECT * FROM Person WHERE GovId = ?",
            SYSTEM.newString("TOYENC486FH"));
    // Check if there is a result
    if (!result.iterator().hasNext()) {
        IonStruct person = SYSTEM.newEmptyStruct();
        person.put("GovId").newString("TOYENC486FH");
```

```
        person.put("FirstName").newString("Brent");
        // Insert the document
        txn.execute("INSERT INTO Person ?", person);
    }
});
```

> **Note**
>
> Dans cet exemple, nous recommandons de disposer d'un index sur le`GovId` terrain afin d'optimiser les performances. Si aucun index n'est activé`GovId`, les instructions peuvent présenter une latence plus importante et peuvent également entraîner des exceptions de conflit OCC ou des délais de transaction.

Utilisation d'Amazon Ion

Vous disposez de plusieurs options pour traiter les données Amazon Ion dans QLDB. Vous pouvez utiliser les méthodes intégrées de la bibliothèque Ion pour créer et modifier des documents de manière flexible selon vos besoins. Vous pouvez également utiliser le module de format de données Jackson de FasterXML pour Ion afin de mapper des documents Ion sur de vieux modèles d'objets Java (POJO).

Les sections suivantes fournissent des exemples de code de traitement de données ioniques utilisant les deux techniques.

Table des matières

Importation des packages Ion

Ajoutez l'artefact ion-java en tant que dépendance dans votre projet Java.

Gradle

```
dependencies {
```

```
    compile group: 'com.amazon.ion', name: 'ion-java', version: '1.6.1'
}
```

Maven

```
<dependencies>
  <dependency>
    <groupId>com.amazon.ion</groupId>
    <artifactId>ion-java</artifactId>
    <version>1.6.1</version>
  </dependency>
</dependencies>
```

Importez les packages Ion suivants.

```
import com.amazon.ion.IonStruct;
import com.amazon.ion.IonSystem;
import com.amazon.ion.system.IonSystemBuilder;
```

Utilisation du mappeur Jackson

Ajoutez l'artefact jackson-dataformat-ionen tant que dépendance dans votre projet Java. QLDB nécessite une version`2.10.0` ou une version ultérieure.

Gradle

```
dependencies {
    compile group: 'com.fasterxml.jackson.dataformat', name: 'jackson-dataformat-
ion', version: '2.10.0'
}
```

Maven

```
<dependencies>
  <dependency>
    <groupId>com.fasterxml.jackson.dataformat</groupId>
    <artifactId>jackson-dataformat-ion</artifactId>
    <version>2.10.0</version>
  </dependency>
</dependencies>
```

Importez les packages Ion suivants.

```
import com.amazon.ion.IonReader;
import com.amazon.ion.IonStruct;
import com.amazon.ion.system.IonReaderBuilder;
import com.amazon.ion.system.IonSystemBuilder;

import com.fasterxml.jackson.dataformat.ion.IonObjectMapper;
import com.fasterxml.jackson.dataformat.ion.ionvalue.IonValueMapper;
```

Ion

```
IonSystem SYSTEM = IonSystemBuilder.standard().build();
```

Utilisation du mappeur Jackson

```
IonObjectMapper MAPPER = new IonValueMapper(IonSystemBuilder.standard().build());
```

Créer des objets Ion

L'exemple de code suivant crée un objet Ion à l'aide de l'`IonStruct`interface et de ses méthodes intégrées.

```
IonStruct ionStruct = SYSTEM.newEmptyStruct();

ionStruct.put("GovId").newString("TOYENC486FH");
ionStruct.put("FirstName").newString("Brent");

System.out.println(ionStruct.toPrettyString()); // prints a nicely formatted copy of
 ionStruct
```

Utilisation du mappeur Jackson

Supposons que vous ayez une classe de modèle mappée en JSON nomméePerson comme suit.

```
import com.fasterxml.jackson.annotation.JsonCreator;
import com.fasterxml.jackson.annotation.JsonProperty;

public static class Person {
    private final String firstName;
```

```
    private final String govId;

    @JsonCreator
    public Person(@JsonProperty("FirstName") final String firstName,
                  @JsonProperty("GovId") final String govId) {
        this.firstName = firstName;
        this.govId = govId;
    }

    @JsonProperty("FirstName")
    public String getFirstName() {
        return firstName;
    }

    @JsonProperty("GovId")
    public String getGovId() {
        return govId;
    }
}
```

L'exemple de code suivant crée un`IonStruct` objet à partir d'une instance de`Person`.

```
IonStruct ionStruct = (IonStruct) MAPPER.writeValueAsIonValue(new Person("Brent",
 "TOYENC486FH"));
```

Lire des objets Ion

L'exemple de code suivant imprime chaque champ de l'`ionStruct`instance.

```
// ionStruct is an instance of IonStruct
System.out.println(ionStruct.get("GovId")); // prints TOYENC486FH
System.out.println(ionStruct.get("FirstName")); // prints Brent
```

Utilisation du mappeur Jackson

L'exemple de code suivant lit un`IonStruct` objet et le mappe à une instance de`Person`.

```
// ionStruct is an instance of IonStruct
IonReader reader = IonReaderBuilder.standard().build(ionStruct);
Person person = MAPPER.readValue(reader, Person.class);
System.out.println(person.getFirstName()); // prints Brent
```

```
System.out.println(person.getGovId()); // prints TOYENC486FH
```

Pour plus d'informations sur l'utilisation d'Ion, consultez la documentation Amazon Ion sur GitHub. Pour plus d'exemples de code illustrant l'utilisation d'Ion dans QLDB, consultezUtilisation des types de données Amazon Ion dans Amazon QLDB.

Pilote Amazon QLDB pour .NET

Pour utiliser les données de votre registre, vous pouvez vous connecter à Amazon QLDB depuis votre application Microsoft .NET à l'aide d'un piloteAWS fourni. Le pilote cible .NET Standard 2.0. Plus précisément, il prend en charge .NET Core (LTS) 2.1+ et .NET Framework 4.5.2+. Pour plus d'informations sur la compatibilité, consultez .NET Standard sur le site Microsoft Docs.

Nous vous recommandons vivement d'utiliser le mappeur d'objets Ion pour éviter complètement de devoir effectuer une conversion manuelle entre les types Amazon Ion et les types C# natifs.

Les rubriques suivantes décrivent comment démarrer avec le pilote QLDB Driver pour .NET.

Rubriques

- Ressources pour les conducteurs ressources
- Prérequis
- Installation
- Pilote Amazon QLDB pour .NET — Tutoriel de démarrage rapide
- Pilote Amazon QLDB pour .NET — Guide de référence du livre de recettes

Ressources pour les conducteurs ressources

Pour plus d'informations sur les fonctionnalités prises en charge par le pilote .NET, consultez les ressources suivantes :

- Référence d'API
- Code source du pilote (GitHub)
- Exemple de code source d'application (GitHub)
- Livre de recettes Amazon Ion
- Cartographe d'objets ioniques (GitHub)

Prérequis

Avant de commencer à utiliser le pilote QLDB Driver pour .NET, vous devez effectuer les opérations suivantes :

1. Suivez les instructions deAWS configuration dansAccès à Amazon QLDB. Cela inclut les éléments suivants :

    1. S'inscrire àAWS.
    2. Créer un utilisateur avec les autorisations QLDB appropriées.
    3. Accorder un accès par programmation à des fins de développement.
2. Téléchargez et installez le SDK .NET Core version 2.1 ou ultérieure à partir du site de téléchargements Microsoft .NET.
3. (Facultatif) Installez l'environnement de développement intégré (IDE) de votre choix, tel que Visual Studio, Visual Studio pour Mac ou Visual Studio Code. Vous pouvez les télécharger depuis le site Microsoft Visual Studio.
4. Configurez votre environnement de développement pour AWS SDK for .NET:

    1. Configurez vosAWS informations d'identification. Nous vous recommandons de créer un fichier d'informations d'identification partagées.

        Pour obtenir des instructions, consultez la section Configuration desAWS informations d'identification à l'aide d'un fichier d'informations d'identification dans le Guide duAWS SDK for .NET développeur.
    2. Définissez votre valeur par défautRégion AWS. Pour savoir comment procéder, consultez la section Région AWSSélection.

        Pour obtenir la liste complète des régions disponibles, consultez les points de terminaison et les quotas Amazon QLDB dans le Références générales AWS.

Vous pouvez ensuite configurer un exemple d'application de base et exécuter des exemples de code abrégé, ou vous pouvez installer le pilote dans un projet .NET existant.

- Pour installer le pilote QLDB et leAWS SDK for .NET dans un projet existant, passez àInstallation.
- Pour configurer un projet et exécuter des exemples de code abrégé illustrant les transactions de données de base sur un registre, consultez leDdémarrage rapide.

Installation

Utilisez le gestionnaire de NuGet packages pour installer le pilote QLDB pour .NET. Nous vous recommandons d'utiliser Visual Studio ou un IDE de votre choix pour ajouter des dépendances au projet. Le nom du package de pilotes est Amazon.QLDB.Driver.

Par exemple, dans Visual Studio, ouvrez la consoleNuGet du gestionnaire de packages dans le menu Outils. À l'PM>invite, entrez la commande suivante.

```
PM> Install-Package Amazon.QLDB.Driver
```

L'installation du pilote installe également ses dépendances, notamment les packages Amazon IonAWS SDK for .NET et Amazon.

Installation du mappeur d'objets Ion

La version 1.3.0 du pilote QLDB pour .NET prend en charge l'acceptation et le renvoi de types de données C# natifs sans qu'il soit nécessaire de travailler avec Amazon Ion. Pour utiliser cette fonctionnalité, ajoutez le package suivant à votre projet.

- Amazon.QLDB.Driver.Serialization : bibliothèque capable de mapper des valeurs Ion à de simples objets CLR (POCO) en C#, et inversement. Ce mappeur d'objets Ion permet à votre application d'interagir directement avec des types de données C# natifs sans avoir à travailler avec Ion. Pour un bref guide sur l'utilisation de cette bibliothèque, consultez le fichier Serialization.md dans le GitHub référentielawslabs/amazon-qldb-driver-dotnet.

Pour installer le package, entrez la commande suivante.

```
PM> Install-Package Amazon.QLDB.Driver.Serialization
```

Pour des exemples de code courts expliquant comment exécuter des transactions de données de base sur un registre, consultez leRéférence de livre de cuisine.

Pilote Amazon QLDB pour .NET — Tutoriel de démarrage rapide

Dans ce tutoriel, vous apprendrez comment configurer une application simple à l'aide du pilote Amazon QLDB pour .NET. Ce guide comprend les étapes d'installation du pilote et des exemples de code court des opérations de base de création, lecture, mise à jour et suppression (CRUD).

Rubriques

Prérequis

Avant de commencer, veillez à exécuter les actions suivantes :

1. Si vous ne l'avez pas déjà fait, complétez lePrérequis pour le pilote .NET, si vous ne l'avez pas déjà fait. Cela inclut l'inscriptionAWS, l'octroi d'un accès programmatique pour le développement et l'installation du SDK .NET Core.
2. Créez un registre nomméquick-start.

   Pour savoir comment créer un registre, consultezOpérations de base pour les registres Amazon QLDB ouÉtape 1 : Créer un nouveau registre dans Prise en main de la console.

Étape 1 : Configurer votre projet

Tout d'abord, configurez votre projet .NET.

1. Pour créer et exécuter un modèle d'application, entrez lesdotnet commandes suivantes sur un terminal tel que bash ou Command Prompt. PowerShell

   ```
   $ dotnet new console --output Amazon.QLDB.QuickStartGuide
   $ dotnet run --project Amazon.QLDB.QuickStartGuide
   ```

   Ce modèle crée un dossier nomméAmazon.QLDB.QuickStartGuide. Dans ce dossier, il crée un projet portant le même nom et un fichier nomméProgram.cs. Le programme contient du code qui affiche la sortieHello World!.

2. Utilisez le gestionnaire de NuGet packages pour installer le pilote QLDB pour .NET. Nous vous recommandons d'utiliser Visual Studio ou un IDE de votre choix pour ajouter des dépendances à votre projet. Le nom du package de pilotes est Amazon.QLDB.Driver.

   - Par exemple, dans Visual Studio, ouvrez la consoleNuGet du gestionnaire de packages dans le menu Outils. À l'PM>invite, entrez la commande suivante.

     ```
     PM> Install-Package Amazon.QLDB.Driver
     ```

   - Vous pouvez également saisir les commandes suivantes sur votre terminal.

     ```
     $ cd Amazon.QLDB.QuickStartGuide
     $ dotnet add package Amazon.QLDB.Driver
     ```

   L'installation du pilote installe également ses dépendances, notamment les bibliothèques Amazon Ion AWS SDK for .NETet Amazon.

3. Installez la bibliothèque de sérialisation du pilote.

   ```
   PM> Install-Package Amazon.QLDB.Driver.Serialization
   ```

4. Ouvrez le fichier `Program.cs`.

   Ajoutez ensuite progressivement les exemples de code dans les étapes suivantes pour essayer certaines opérations CRUD de base. Vous pouvez également ignorer le step-by-step didacticiel et exécuter l'application complète.

> **Note**
>
> - Choix entre des API synchrones et asynchrones : le pilote fournit des API synchrones et asynchrones. Pour les applications à forte demande qui gèrent plusieurs demandes sans blocage, nous recommandons d'utiliser les API asynchrones pour améliorer les performances. Le pilote propose des API synchrones pour plus de commodité pour les bases de code existantes écrites de manière synchrone.
>
>   Ce didacticiel inclut des exemples de code synchrone et asynchrone. Pour plus d'informations sur les API, consultez lesAsyncQldbDriver interfaces IQldbDriver et I dans la documentation de l'API.

- Traitement des données Amazon Ion : ce didacticiel fournit des exemples de code permettant de traiter des données Amazon Ion à l'aide du mappeur d'objets Ion par défaut. QLDB a introduit le mappeur d'objets Ion dans la version 1.3.0 du pilote .NET. Le cas échéant, ce didacticiel fournit également des exemples de code utilisant la bibliothèque Ion standard comme alternative. Pour en savoir plus, consultez Utilisation d'Amazon Ion.

Étape 2 : initialiser le pilote

Initialisez une instance du pilote qui se connecte au registre nomméquick-start. Ajoutez le code suivant à votreProgram.cs fichier.

Async

```
using Amazon.QLDB.Driver;
using Amazon.QLDB.Driver.Generic;
using Amazon.QLDB.Driver.Serialization;

namespace Amazon.QLDB.QuickStartGuide
{
    class Program
    {
        public class Person
        {
            public string FirstName { get; set; }

            public string LastName { get; set; }

            public int Age { get; set; }

            public override string ToString()
            {
                return FirstName + ", " + LastName + ", " + Age.ToString();
            }
        }

        static async Task Main(string[] args)
        {
            Console.WriteLine("Create the async QLDB driver");
            IAsyncQldbDriver driver = AsyncQldbDriver.Builder()
                .WithLedger("quick-start")
                .WithSerializer(new ObjectSerializer())
```

```
                .Build();
        }
    }
}
```

Sync

```
using Amazon.QLDB.Driver;
using Amazon.QLDB.Driver.Generic;
using Amazon.QLDB.Driver.Serialization;

namespace Amazon.QLDB.QuickStartGuide
{
    class Program
    {
        public class Person
        {
            public string FirstName { get; set; }

            public string LastName { get; set; }

            public int Age { get; set; }

            public override string ToString()
            {
                return FirstName + ", " + LastName + ", " + Age.ToString();
            }
        }

        static void Main(string[] args)
        {
            Console.WriteLine("Create the sync QLDB driver");
            IQldbDriver driver = QldbDriver.Builder()
                .WithLedger("quick-start")
                .WithSerializer(new ObjectSerializer())
                .Build();
        }
    }
}
```

Utilisation de la bibliothèque Ion

Async

```
using System;
using System.Threading.Tasks;
using Amazon.IonDotnet.Tree;
using Amazon.IonDotnet.Tree.Impl;
using Amazon.QLDB.Driver;
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;

namespace Amazon.QLDB.QuickStartGuide
{
    class Program
    {
        static IValueFactory valueFactory = new ValueFactory();

        static async Task Main(string[] args)
        {
            Console.WriteLine("Create the async QLDB driver");
            IAsyncQldbDriver driver = AsyncQldbDriver.Builder()
                .WithLedger("quick-start")
                .Build();
        }
    }
}
```

Sync

```
using System;
using Amazon.IonDotnet.Tree;
using Amazon.IonDotnet.Tree.Impl;
using Amazon.QLDB.Driver;

namespace Amazon.QLDB.QuickStartGuide
{
    class Program
    {
        static IValueFactory valueFactory = new ValueFactory();

        static void Main(string[] args)
        {
            Console.WriteLine("Create the sync QLDB Driver");
```

```
            IQldbDriver driver = QldbDriver.Builder()
                .WithLedger("quick-start")
                .Build();
        }
    }
}
```

Étape 3 : Créez une table et un index

Pour la suite de ce didacticiel, jusqu'à l'étape 6, vous devez ajouter les exemples de code suivants à l'exemple de code précédent.

Au cours de cette étape, le code suivant montre comment exécuter`CREATE TABLE` les`CREATE INDEX` instructions and. Il crée une table nommée`Person` et un index pour le`firstName` champ de cette table. Les index sont nécessaires pour optimiser les performances des requêtes et contribuer à limiter les exceptions de conflit liées au contrôle optimiste de la concurrence (OCC).

Async

```
Console.WriteLine("Creating the table and index");

// Creates the table and the index in the same transaction.
// Note: Any code within the lambda can potentially execute multiple times due to
 retries.
// For more information, see: https://docs.aws.amazon.com/qldb/latest/
developerguide/driver-retry-policy
await driver.Execute(async txn =>
{
    await txn.Execute("CREATE TABLE Person");
    await txn.Execute("CREATE INDEX ON Person(firstName)");
});
```

Sync

```
Console.WriteLine("Creating the tables and index");

// Creates the table and the index in the same transaction.
// Note: Any code within the lambda can potentially execute multiple times due to
 retries.
// For more information, see: https://docs.aws.amazon.com/qldb/latest/
developerguide/driver-retry-policy
```

```
driver.Execute(txn =>
{
    txn.Execute("CREATE TABLE Person");
    txn.Execute("CREATE INDEX ON Person(firstName)");
});
```

Étape 4 : Insertion d'un document

L'exemple de code suivant montre comment exécuter une`INSERT` instruction. QLDB prend en charge le langage de requête PartiQL (compatible SQL) et le format de données Amazon Ion (surensemble de JSON).

Ajoutez le code suivant pour insérer un document dans le`Person` tableau.

Async

```
Console.WriteLine("Inserting a document");

Person myPerson = new Person {
    FirstName = "John",
    LastName = "Doe",
    Age = 32
};

await driver.Execute(async txn =>
{
    IQuery<Person> myQuery = txn.Query<Person>("INSERT INTO Person ?", myPerson);
    await txn.Execute(myQuery);
});
```

Sync

```
Console.WriteLine("Inserting a document");

Person myPerson = new Person {
    FirstName = "John",
    LastName = "Doe",
    Age = 32
};

driver.Execute(txn =>
```

```
{
    IQuery<Person> myQuery = txn.Query<Person>("INSERT INTO Person ?", myPerson);
    txn.Execute(myQuery);
});
```

Utilisation de la bibliothèque Ion

Async

```
Console.WriteLine("Inserting a document");

// This is one way of creating Ion values. We can also use an IonLoader.
// For more details, see: https://docs.aws.amazon.com/qldb/latest/developerguide/
driver-cookbook-dotnet.html#cookbook-dotnet.ion
IIonValue ionPerson = valueFactory.NewEmptyStruct();
ionPerson.SetField("firstName", valueFactory.NewString("John"));
ionPerson.SetField("lastName", valueFactory.NewString("Doe"));
ionPerson.SetField("age", valueFactory.NewInt(32));

await driver.Execute(async txn =>
{
    await txn.Execute("INSERT INTO Person ?", ionPerson);
});
```

Sync

```
Console.WriteLine("Inserting a document");

// This is one way of creating Ion values, we can also use an IonLoader.
// For more details, see: https://docs.aws.amazon.com/qldb/latest/developerguide/
driver-cookbook-dotnet.html#cookbook-dotnet.ion
IIonValue ionPerson = valueFactory.NewEmptyStruct();
ionPerson.SetField("firstName", valueFactory.NewString("John"));
ionPerson.SetField("lastName", valueFactory.NewString("Doe"));
ionPerson.SetField("age", valueFactory.NewInt(32));

driver.Execute(txn =>
{
    txn.Execute("INSERT INTO Person ?", ionPerson);
});
```

> **Tip**
>
> Pour insérer plusieurs documents à l'aide d'une seuleINSERT instruction, vous pouvez transmettre un paramètre de type liste Ion à l'instruction comme suit.
>
> ```
> // people is an Ion list
> txn.Execute("INSERT INTO Person ?", people);
> ```
>
> Vous ne placez pas l'espace réservé à la variable (?) entre crochets (<<...>>) lorsque vous transmettez une liste Ion. Dans les instructions manuelles de PartiQL, les crochets doubles indiquent une collection non ordonnée appelée sac.

Étape 5 : interroger le document

L'exemple de code suivant montre comment exécuter uneSELECT instruction.

Ajoutez le code suivant qui interroge un document à partir duPerson tableau.

Async

```
Console.WriteLine("Querying the table");

// The result from driver.Execute() is buffered into memory because once the
// transaction is committed, streaming the result is no longer possible.
IAsyncResult<Person> selectResult = await driver.Execute(async txn =>
{
    IQuery<Person> myQuery = txn.Query<Person>("SELECT * FROM Person WHERE FirstName
 = ?", "John");
    return await txn.Execute(myQuery);
});

await foreach (Person person in selectResult)
{
    Console.WriteLine(person);
    // John, Doe, 32
}
```

Sync

```
Console.WriteLine("Querying the table");
```

```
// The result from driver.Execute() is buffered into memory because once the
// transaction is committed, streaming the result is no longer possible.
IResult<Person> selectResult = driver.Execute(txn =>
{
    IQuery<Person> myQuery = txn.Query<Person>("SELECT * FROM Person WHERE FirstName
 = ?", "John");
    return txn.Execute(myQuery);
});

foreach (Person person in selectResult)
{
    Console.WriteLine(person);
    // John, Doe, 32
}
```

Utilisation de la bibliothèque Ion

Async

```
Console.WriteLine("Querying the table");

IIonValue ionFirstName = valueFactory.NewString("John");

// The result from driver.Execute() is buffered into memory because once the
// transaction is committed, streaming the result is no longer possible.
IAsyncResult selectResult = await driver.Execute(async txn =>
{
    return await txn.Execute("SELECT * FROM Person WHERE firstName = ?",
 ionFirstName);
});

await foreach (IIonValue row in selectResult)
{
    Console.WriteLine(row.GetField("firstName").StringValue);
    Console.WriteLine(row.GetField("lastName").StringValue);
    Console.WriteLine(row.GetField("age").IntValue);
}
```

Sync

```
Console.WriteLine("Querying the table");
```

```
IIonValue ionFirstName = valueFactory.NewString("John");

// The result from driver.Execute() is buffered into memory because once the
// transaction is committed, streaming the result is no longer possible.
IResult selectResult = driver.Execute(txn =>
{
    return txn.Execute("SELECT * FROM Person WHERE firstName = ?", ionFirstName);
});

foreach (IIonValue row in selectResult)
{
    Console.WriteLine(row.GetField("firstName").StringValue);
    Console.WriteLine(row.GetField("lastName").StringValue);
    Console.WriteLine(row.GetField("age").IntValue);
}
```

Cet exemple utilise un point d'interrogation (?) comme espace réservé variable pour transmettre les informations du document à l'instruction. Lorsque vous utilisez des espaces réservés, vous devez transmettre une valeur de type`IonValue`.

Étape 6 : Mettre le document à jour

L'exemple de code suivant montre comment exécuter une`UPDATE` instruction.

1. Ajoutez le code suivant pour mettre à jour un document dans le`Person` tableau en le mettant à jour`age` vers 42.

   Async

   ```
   Console.WriteLine("Updating the document");

   await driver.Execute(async txn =>
   {
       IQuery<Person> myQuery = txn.Query<Person>("UPDATE Person SET Age = ? WHERE
    FirstName = ?", 42, "John");
       await txn.Execute(myQuery);
   });
   ```

Sync

```
Console.WriteLine("Updating the document");

driver.Execute(txn =>
{
    IQuery<Person> myQuery = txn.Query<Person>("UPDATE Person SET Age = ? WHERE
 FirstName = ?", 42, "John");
    txn.Execute(myQuery);
});
```

2. Interrogez à nouveau le document pour voir la valeur mise à jour.

Async

```
Console.WriteLine("Querying the table for the updated document");

IAsyncResult<Person> updateResult = await driver.Execute(async txn =>
{
    IQuery<Person> myQuery = txn.Query<Person>("SELECT * FROM Person WHERE
 FirstName = ?", "John");
    return await txn.Execute(myQuery);
});

await foreach (Person person in updateResult)
{
    Console.WriteLine(person);
    // John, Doe, 42
}
```

Sync

```
Console.WriteLine("Querying the table for the updated document");

IResult<Person> updateResult = driver.Execute(txn =>
{
    IQuery<Person> myQuery = txn.Query<Person>("SELECT * FROM Person WHERE
 FirstName = ?", "John");
    return txn.Execute(myQuery);
});

foreach (Person person in updateResult)
```

```
{
    Console.WriteLine(person);
    // John, Doe, 42
}
```

3. Pour exécuter l'application, entrez la commande suivante depuis le répertoire parent du répertoire de votre`Amazon.QLDB.QuickStartGuide` projet.

```
$ dotnet run --project Amazon.QLDB.QuickStartGuide
```

Utilisation de la bibliothèque Ion

1. Ajoutez le code suivant pour mettre à jour un document dans le`Person` tableau en le mettant à jour`age` vers 42.

   Async

```
Console.WriteLine("Updating the document");

IIonValue ionIntAge = valueFactory.NewInt(42);
IIonValue ionFirstName2 = valueFactory.NewString("John");

await driver.Execute(async txn =>
{
    await txn.Execute("UPDATE Person SET age = ? WHERE firstName = ?",
 ionIntAge, ionFirstName2);
});
```

   Sync

```
Console.WriteLine("Updating a document");

IIonValue ionIntAge = valueFactory.NewInt(42);
IIonValue ionFirstName2 = valueFactory.NewString("John");

driver.Execute(txn =>
{
    txn.Execute("UPDATE Person SET age = ? WHERE firstName = ?", ionIntAge,
 ionFirstName2);
});
```

2. Interrogez à nouveau le document pour voir la valeur mise à jour.

   Async

```
Console.WriteLine("Querying the table for the updated document");

IIonValue ionFirstName3 = valueFactory.NewString("John");

IAsyncResult updateResult = await driver.Execute(async txn =>
{
    return await txn.Execute("SELECT * FROM Person WHERE firstName = ?",
 ionFirstName3 );
});

await foreach (IIonValue row in updateResult)
{
    Console.WriteLine(row.GetField("firstName").StringValue);
    Console.WriteLine(row.GetField("lastName").StringValue);
    Console.WriteLine(row.GetField("age").IntValue);
}
```

   Sync

```
Console.WriteLine("Querying the table for the updated document");

IIonValue ionFirstName3 = valueFactory.NewString("John");

IResult updateResult = driver.Execute(txn =>
{
    return txn.Execute("SELECT * FROM Person WHERE firstName = ?",
 ionFirstName3);
});

foreach (IIonValue row in updateResult)
{
    Console.WriteLine(row.GetField("firstName").StringValue);
    Console.WriteLine(row.GetField("lastName").StringValue);
    Console.WriteLine(row.GetField("age").IntValue);
}
```

3. Pour exécuter l'application, entrez la commande suivante depuis le répertoire parent du répertoire de votre`Amazon.QLDB.QuickStartGuide` projet.

```
$ dotnet run --project Amazon.QLDB.QuickStartGuide
```

Exécution de l'application complète

L'exemple de code suivant représente la version complète de l'`Program.cs`application. Au lieu de suivre les étapes précédentes individuellement, vous pouvez également copier et exécuter cet exemple de code du début à la fin. Cette application montre certaines opérations CRUD de base sur le registre nomméquick-start.

> **Note**
>
> Avant d'exécuter ce code, assurez-vous qu'aucune table active n'est déjà nomméePerson dans lequick-start registre.

Async

```
using Amazon.QLDB.Driver;
using Amazon.QLDB.Driver.Generic;
using Amazon.QLDB.Driver.Serialization;

namespace Amazon.QLDB.QuickStartGuide
{
    class Program
    {
        public class Person
        {
            public string FirstName { get; set; }

            public string LastName { get; set; }

            public int Age { get; set; }

            public override string ToString()
            {
                return FirstName + ", " + LastName + ", " + Age.ToString();
            }
        }

        static async Task Main(string[] args)
```

```
        {
            Console.WriteLine("Create the async QLDB driver");
            IAsyncQldbDriver driver = AsyncQldbDriver.Builder()
                .WithLedger("quick-start")
                .WithSerializer(new ObjectSerializer())
                .Build();

            Console.WriteLine("Creating the table and index");

            // Creates the table and the index in the same transaction.
            // Note: Any code within the lambda can potentially execute multiple
 times due to retries.
            // For more information, see: https://docs.aws.amazon.com/qldb/latest/
developerguide/driver-retry-policy
            await driver.Execute(async txn =>
            {
                await txn.Execute("CREATE TABLE Person");
                await txn.Execute("CREATE INDEX ON Person(firstName)");
            });

            Console.WriteLine("Inserting a document");

            Person myPerson = new Person {
                FirstName = "John",
                LastName = "Doe",
                Age = 32
            };

            await driver.Execute(async txn =>
            {
                IQuery<Person> myQuery = txn.Query<Person>("INSERT INTO Person ?",
 myPerson);
                await txn.Execute(myQuery);
            });

            Console.WriteLine("Querying the table");

            // The result from driver.Execute() is buffered into memory because once
 the
            // transaction is committed, streaming the result is no longer possible.
            IAsyncResult<Person> selectResult = await driver.Execute(async txn =>
            {
                IQuery<Person> myQuery = txn.Query<Person>("SELECT * FROM Person
 WHERE FirstName = ?", "John");
```

```
                return await txn.Execute(myQuery);
            });

            await foreach (Person person in selectResult)
            {
                Console.WriteLine(person);
                // John, Doe, 32
            }

            Console.WriteLine("Updating the document");

            await driver.Execute(async txn =>
            {
                IQuery<Person> myQuery = txn.Query<Person>("UPDATE Person SET Age
 = ? WHERE FirstName = ?", 42, "John");
                await txn.Execute(myQuery);
            });

            Console.WriteLine("Querying the table for the updated document");

            IAsyncResult<Person> updateResult = await driver.Execute(async txn =>
            {
                IQuery<Person> myQuery = txn.Query<Person>("SELECT * FROM Person
 WHERE FirstName = ?", "John");
                return await txn.Execute(myQuery);
            });

            await foreach (Person person in updateResult)
            {
                Console.WriteLine(person);
                // John, Doe, 42
            }
        }
    }
}
```

Sync

```
using Amazon.QLDB.Driver;
using Amazon.QLDB.Driver.Generic;
using Amazon.QLDB.Driver.Serialization;

namespace Amazon.QLDB.QuickStartGuide
```

```
{
    class Program
    {
        public class Person
        {
            public string FirstName { get; set; }

            public string LastName { get; set; }

            public int Age { get; set; }

            public override string ToString()
            {
                return FirstName + ", " + LastName + ", " + Age.ToString();
            }
        }

        static void Main(string[] args)
        {
            Console.WriteLine("Create the sync QLDB driver");
            IQldbDriver driver = QldbDriver.Builder()
                .WithLedger("quick-start")
                .WithSerializer(new ObjectSerializer())
                .Build();

            Console.WriteLine("Creating the table and index");

            // Creates the table and the index in the same transaction.
            // Note: Any code within the lambda can potentially execute multiple
 times due to retries.
            // For more information, see: https://docs.aws.amazon.com/qldb/latest/
developerguide/driver-retry-policy
            driver.Execute(txn =>
            {
                txn.Execute("CREATE TABLE Person");
                txn.Execute("CREATE INDEX ON Person(firstName)");
            });

            Console.WriteLine("Inserting a document");

            Person myPerson = new Person {
                FirstName = "John",
                LastName = "Doe",
                Age = 32
```

```
            };

            driver.Execute(txn =>
            {
                IQuery<Person> myQuery = txn.Query<Person>("INSERT INTO Person ?",
myPerson);
                txn.Execute(myQuery);
            });

            Console.WriteLine("Querying the table");

            // The result from driver.Execute() is buffered into memory because once
the
            // transaction is committed, streaming the result is no longer possible.
            IResult<Person> selectResult = driver.Execute(txn =>
            {
                IQuery<Person> myQuery = txn.Query<Person>("SELECT * FROM Person
WHERE FirstName = ?", "John");
                return txn.Execute(myQuery);
            });

            foreach (Person person in selectResult)
            {
                Console.WriteLine(person);
                // John, Doe, 32
            }

            Console.WriteLine("Updating the document");

            driver.Execute(txn =>
            {
                IQuery<Person> myQuery = txn.Query<Person>("UPDATE Person SET Age
= ? WHERE FirstName = ?", 42, "John");
                txn.Execute(myQuery);
            });

            Console.WriteLine("Querying the table for the updated document");

            IResult<Person> updateResult = driver.Execute(txn =>
            {
                IQuery<Person> myQuery = txn.Query<Person>("SELECT * FROM Person
WHERE FirstName = ?", "John");
                return txn.Execute(myQuery);
            });
```

```
            foreach (Person person in updateResult)
            {
                Console.WriteLine(person);
                // John, Doe, 42
            }

        }
    }
}
```

Utilisation de la bibliothèque Ion

Async

```
using System;
using System.Threading.Tasks;
using Amazon.IonDotnet.Tree;
using Amazon.IonDotnet.Tree.Impl;
using Amazon.QLDB.Driver;
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;

namespace Amazon.QLDB.QuickStartGuide
{
    class Program
    {
        static IValueFactory valueFactory = new ValueFactory();

        static async Task Main(string[] args)
        {
            Console.WriteLine("Create the async QLDB driver");
            IAsyncQldbDriver driver = AsyncQldbDriver.Builder()
                .WithLedger("quick-start")
                .Build();

            Console.WriteLine("Creating the table and index");

            // Creates the table and the index in the same transaction.
            // Note: Any code within the lambda can potentially execute multiple
 times due to retries.
            // For more information, see: https://docs.aws.amazon.com/qldb/latest/
developerguide/driver-retry-policy
```

```
            await driver.Execute(async txn =>
            {
                await txn.Execute("CREATE TABLE Person");
                await txn.Execute("CREATE INDEX ON Person(firstName)");
            });

            Console.WriteLine("Inserting a document");

            // This is one way of creating Ion values. We can also use an IonLoader.
            // For more details, see: https://docs.aws.amazon.com/qldb/latest/
developerguide/driver-cookbook-dotnet.html#cookbook-dotnet.ion
            IIonValue ionPerson = valueFactory.NewEmptyStruct();
            ionPerson.SetField("firstName", valueFactory.NewString("John"));
            ionPerson.SetField("lastName", valueFactory.NewString("Doe"));
            ionPerson.SetField("age", valueFactory.NewInt(32));

            await driver.Execute(async txn =>
            {
                await txn.Execute("INSERT INTO Person ?", ionPerson);
            });

            Console.WriteLine("Querying the table");

            IIonValue ionFirstName = valueFactory.NewString("John");

            // The result from driver.Execute() is buffered into memory because once
 the
            // transaction is committed, streaming the result is no longer possible.
            IAsyncResult selectResult = await driver.Execute(async txn =>
            {
                return await txn.Execute("SELECT * FROM Person WHERE firstName = ?",
 ionFirstName);
            });

            await foreach (IIonValue row in selectResult)
            {
                Console.WriteLine(row.GetField("firstName").StringValue);
                Console.WriteLine(row.GetField("lastName").StringValue);
                Console.WriteLine(row.GetField("age").IntValue);
            }

            Console.WriteLine("Updating the document");

            IIonValue ionIntAge = valueFactory.NewInt(42);
```

```
            IIonValue ionFirstName2 = valueFactory.NewString("John");

            await driver.Execute(async txn =>
            {
                await txn.Execute("UPDATE Person SET age = ? WHERE firstName = ?",
 ionIntAge, ionFirstName2);
            });

            Console.WriteLine("Querying the table for the updated document");

            IIonValue ionFirstName3 = valueFactory.NewString("John");

            IAsyncResult updateResult = await driver.Execute(async txn =>
            {
                return await txn.Execute("SELECT * FROM Person WHERE firstName = ?",
 ionFirstName3);
            });

            await foreach (IIonValue row in updateResult)
            {
                Console.WriteLine(row.GetField("firstName").StringValue);
                Console.WriteLine(row.GetField("lastName").StringValue);
                Console.WriteLine(row.GetField("age").IntValue);
            }
        }
    }
}
```

Sync

```
using System;
using Amazon.IonDotnet.Tree;
using Amazon.IonDotnet.Tree.Impl;
using Amazon.QLDB.Driver;

namespace Amazon.QLDB.QuickStartGuide
{
    class Program
    {
        static IValueFactory valueFactory = new ValueFactory();

        static void Main(string[] args)
        {
```

```
            Console.WriteLine("Create the sync QLDB Driver");
            IQldbDriver driver = QldbDriver.Builder()
                .WithLedger("quick-start")
                .Build();

            Console.WriteLine("Creating the tables and index");

            // Creates the table and the index in the same transaction.
            // Note: Any code within the lambda can potentially execute multiple
 times due to retries.
            // For more information, see: https://docs.aws.amazon.com/qldb/latest/
developerguide/driver-retry-policy
            driver.Execute(txn =>
            {
                txn.Execute("CREATE TABLE Person");
                txn.Execute("CREATE INDEX ON Person(firstName)");
            });

            Console.WriteLine("Inserting a document");

            // This is one way of creating Ion values. We can also use an IonLoader.
            // For more details, see: https://docs.aws.amazon.com/qldb/latest/
developerguide/driver-cookbook-dotnet.html#cookbook-dotnet.ion
            IIonValue ionPerson = valueFactory.NewEmptyStruct();
            ionPerson.SetField("firstName", valueFactory.NewString("John"));
            ionPerson.SetField("lastName", valueFactory.NewString("Doe"));
            ionPerson.SetField("age", valueFactory.NewInt(32));

            driver.Execute(txn =>
            {
                txn.Execute("INSERT INTO Person ?", ionPerson);
            });

            Console.WriteLine("Querying the table");

            IIonValue ionFirstName = valueFactory.NewString("John");

            // The result from driver.Execute() is buffered into memory because once
 the
            // transaction is committed, streaming the result is no longer possible.
            IResult selectResult = driver.Execute(txn =>
            {
                return txn.Execute("SELECT * FROM Person WHERE firstName = ?",
 ionFirstName);
```

```
            });

            foreach (IIonValue row in selectResult)
            {
                Console.WriteLine(row.GetField("firstName").StringValue);
                Console.WriteLine(row.GetField("lastName").StringValue);
                Console.WriteLine(row.GetField("age").IntValue);
            }

            Console.WriteLine("Updating a document");

            IIonValue ionIntAge = valueFactory.NewInt(42);
            IIonValue ionFirstName2 = valueFactory.NewString("John");

            driver.Execute(txn =>
            {
                txn.Execute("UPDATE Person SET age = ? WHERE firstName = ?",
 ionIntAge, ionFirstName2);
            });

            Console.WriteLine("Querying the table for the updated document");

            IIonValue ionFirstName3 = valueFactory.NewString("John");

            IResult updateResult = driver.Execute(txn =>
            {
                return txn.Execute("SELECT * FROM Person WHERE firstName = ?",
 ionFirstName3);
            });

            foreach (IIonValue row in updateResult)
            {
                Console.WriteLine(row.GetField("firstName").StringValue);
                Console.WriteLine(row.GetField("lastName").StringValue);
                Console.WriteLine(row.GetField("age").IntValue);
            }
        }
    }
}
```

Pour exécuter l'application complète, entrez la commande suivante depuis le répertoire parent du répertoire de votre`Amazon.QLDB.QuickStartGuide` projet.

```
$ dotnet run --project Amazon.QLDB.QuickStartGuide
```

Pilote Amazon QLDB pour .NET — Guide de référence du livre de recettes

Ce guide de référence présente les cas d'utilisation courants du pilote Amazon QLDB pour .NET. Il fournit des exemples de code C# montrant comment utiliser le pilote pour exécuter des opérations de création, lecture et mise à jour et suppression (CRUD) de base. Il inclut également des exemples de code pour le traitement des données Amazon Ion. En outre, ce guide met en évidence les meilleures pratiques pour rendre les transactions idempotentes et mettre en œuvre des contraintes d'unicité.

> **Note**
>
> Cette rubrique fournit des exemples de code relatifs au traitement de données Amazon Ion à l'aide du mappeur d'objets Ion par défaut. QLDB a introduit le mappeur d'objets Ion dans la version 1.3.0 du pilote .NET. Le cas échéant, cette rubrique fournit également des exemples de code utilisant la bibliothèque Ion standard comme alternative. Pour en savoir plus, consultez Utilisation d'Amazon Ion.

Table des matières

Importation du pilote

L'exemple de code suivant importe le pilote.

```
using Amazon.QLDB.Driver;
using Amazon.QLDB.Driver.Generic;
using Amazon.QLDB.Driver.Serialization;
```

Utilisation de la bibliothèque Ion

```
using Amazon.QLDB.Driver;
using Amazon.IonDotnet.Builders;
```

Instanciation du pilote

L'exemple de code suivant crée une instance du pilote qui se connecte à un nom de registre spécifié à l'aide des paramètres par défaut.

Async

```
IAsyncQldbDriver driver = AsyncQldbDriver.Builder()
    .WithLedger("vehicle-registration")
    // Add Serialization library
    .WithSerializer(new ObjectSerializer())
    .Build();
```

Sync

```
IQldbDriver driver = QldbDriver.Builder()
    .WithLedger("vehicle-registration")
    // Add Serialization library
    .WithSerializer(new ObjectSerializer())
```

```
    .Build();
```

Utilisation de la bibliothèque Ion

Async

```
IAsyncQldbDriver driver = AsyncQldbDriver.Builder().WithLedger("vehicle-
registration").Build();
```

Sync

```
IQldbDriver driver = QldbDriver.Builder().WithLedger("vehicle-
registration").Build();
```

Opérations CRUD

QLDB exécute des opérations de création, lecture, mise à jour et suppression (CRUD) dans le cadre d'une transaction.

> **Warning**
>
> À titre de bonne pratique, faites de vos transactions d'écriture strictement idempotentes.

Rendre les transactions idempotentes

Nous vous recommandons de rendre les transactions d'écriture idempotentes afin d'éviter tout effet secondaire inattendu en cas de nouvelles tentatives. Une transaction est idempotente si elle peut être exécutée plusieurs fois et produire des résultats identiques à chaque fois.

Prenons l'exemple d'une transaction qui insère un document dans une table nomméePerson. La transaction doit d'abord vérifier si le document existe déjà dans le tableau. Sans cette vérification, le tableau risque de contenir des documents dupliqués.

Supposons que QLDB valide avec succès la transaction côté serveur, mais que le client expire en attendant une réponse. Si la transaction n'est pas idempotente, le même document peut être inséré plusieurs fois en cas de nouvelle tentative.

Utilisation d'index pour éviter l'analyse complète des tables

Nous vous recommandons également d'exécuter des instructions avec une clause de`WHERE` prédicat à l'aide d'un opérateur d'égalité sur un champ indexé ou un identifiant de document, par exemple,`WHERE indexedField = 123` ou`WHERE indexedField IN (456, 789)`. Sans cette recherche indexée, QLDB doit effectuer une analyse des tables, ce qui peut entraîner des délais d'expiration des transactions ou des conflits de contrôle optimiste de la concurrence (OCC).

Pour plus d'informations sur OCC, consultezModèle de mise QLDB simultansimultansimultansimultansimultansimultansimultansimultansimultanéité.

Transactions créées implicitement

La méthode Amazon.QLDB.Driver.IQldbDriver .Execute accepte une fonction Lambda qui reçoit une instance de Amazon.QLDB.Driver. TransactionExecutor, que vous pouvez utiliser pour exécuter des instructions. L'instance de`TransactionExecutor` encapsule une transaction créée implicitement.

Vous pouvez exécuter des instructions dans la fonction Lambda à l'aide de la`Execute` méthode de l'exécuteur de transactions. Le pilote valide implicitement la transaction lorsque la fonction Lambda est renvoyée.

Les sections suivantes montrent comment exécuter des opérations CRUD de base, spécifier une logique de nouvelle tentative personnalisée et implémenter des contraintes d'unicité.

Table des matières

Création de tables

Async

```
IAsyncResult<Table> createResult = await driver.Execute(async txn =>
{
    IQuery<Table> query = txn.Query<Table>("CREATE TABLE Person");
    return await txn.Execute(query);
});

await foreach (var result in createResult)
{
    Console.WriteLine("{ tableId: " + result.TableId + " }");
    // The statement returns the created table ID:
    // { tableId: 4o5Uk09OcjC6PpJpLahceE }
}
```

Sync

```
IResult<Table> createResult = driver.Execute( txn =>
{
    IQuery<Table> query = txn.Query<Table>("CREATE TABLE Person");
    return txn.Execute(query);
});

foreach (var result in createResult)
{
    Console.WriteLine("{ tableId: " + result.TableId + " }");
    // The statement returns the created table ID:
    // { tableId: 4o5Uk09OcjC6PpJpLahceE }
}
```

Utilisation de la bibliothèque Ion

Async

```
// The result from driver.Execute() is buffered into memory because once the
// transaction is committed, streaming the result is no longer possible.
IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("CREATE TABLE Person");
});
```

```
await foreach (IIonValue row in result)
{
    Console.WriteLine(row.ToPrettyString());
    // The statement returns the created table ID:
    // {
    //    tableId: "4o5Uk09OcjC6PpJpLahceE"
    // }
}
```

Sync

```
// The result from driver.Execute() is buffered into memory because once the
// transaction is committed, streaming the result is no longer possible.
IResult result = driver.Execute(txn =>
{
    return txn.Execute("CREATE TABLE Person");
});

foreach (IIonValue row in result)
{
    Console.WriteLine(row.ToPrettyString());
    // The statement returns the created table ID:
    // {
    //    tableId: "4o5Uk09OcjC6PpJpLahceE"
    // }
}
```

Création d'index

Async

```
IAsyncResult<Table> createResult = await driver.Execute(async txn =>
{
    IQuery<Table> query = txn.Query<Table>("CREATE INDEX ON Person(firstName)");
    return await txn.Execute(query);
});

await foreach (var result in createResult)
{
    Console.WriteLine("{ tableId: " + result.TableId + " }");
    // The statement returns the updated table ID:
```

```
    // { tableId: 4o5Uk09OcjC6PpJpLahceE }
}
```

Sync

```
IResult<Table> createResult = driver.Execute(txn =>
{
    IQuery<Table> query = txn.Query<Table>("CREATE INDEX ON Person(firstName)");
    return txn.Execute(query);
});

foreach (var result in createResult)
{
    Console.WriteLine("{ tableId: " + result.TableId + " }");
    // The statement returns the updated table ID:
    // { tableId: 4o5Uk09OcjC6PpJpLahceE }
}
```

Utilisation de la bibliothèque Ion

Async

```
IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("CREATE INDEX ON Person(GovId)");
});

await foreach (IIonValue row in result)
{
    Console.WriteLine(row.ToPrettyString());
    // The statement returns the updated table ID:
    // {
    //    tableId: "4o5Uk09OcjC6PpJpLahceE"
    // }
}
```

Sync

```
IResult result = driver.Execute(txn =>
{
    return txn.Execute("CREATE INDEX ON Person(GovId)");
});
```

```
foreach (IIonValue row in result)
{
    Console.WriteLine(row.ToPrettyString());
    // The statement returns the updated table ID:
    // {
    //    tableId: "4o5Uk09OcjC6PpJpLahceE"
    // }
}
```

Lecture de documents

```
// Assumes that Person table has documents as follows:
// { "GovId": "TOYENC486FH", "FirstName" : "Brent" }
// Person class is defined as follows:
// public class Person
// {
//     public string GovId { get; set; }
//     public string FirstName { get; set; }
//  }

IAsyncResult<Person> result = await driver.Execute(async txn =>
{
    return await txn.Execute(txn.Query<Person>("SELECT * FROM Person WHERE GovId =
 'TOYENC486FH'"));
});

await foreach (Person person in result)
{
    Console.WriteLine(person.GovId); // Prints TOYENC486FH.
    Console.WriteLine(person.FirstName); // Prints Brent.
}
```

> **Note**
>
> Lorsque vous exécutez une requête sans recherche indexée, elle appelle une analyse complète de la table. Dans cet exemple, nous recommandons de disposer d'un index sur le`GovId` terrain afin d'optimiser les performances. Si aucun index n'est activé`GovId`, les requêtes peuvent présenter une latence plus importante et peuvent également entraîner des exceptions de conflit OCC ou des délais de transaction.

Utilisation des paramètres de requête

L'exemple de code suivant utilise un paramètre de requête de type C#.

```
IAsyncResult<Person> result = await driver.Execute(async txn =>
{
    return await txn.Execute(txn.Query<Person>("SELECT * FROM Person WHERE FirstName
 = ?", "Brent"));
});

await foreach (Person person in result)
{
    Console.WriteLine(person.GovId); // Prints TOYENC486FH.
    Console.WriteLine(person.FirstName); // Prints Brent.
}
```

L'exemple de code suivant utilise plusieurs paramètres de requête de type C#.

```
IAsyncResult<Person> result = await driver.Execute(async txn =>
{
    return await txn.Execute(txn.Query<Person>("SELECT * FROM Person WHERE GovId = ?
 AND FirstName = ?", "TOYENC486FH", "Brent"));
});

await foreach (Person person in result)
{
    Console.WriteLine(person.GovId); // Prints TOYENC486FH.
    Console.WriteLine(person.FirstName); // Prints Brent.
}
```

L'exemple de code suivant utilise un tableau de paramètres de requête de type C#.

```
// Assumes that Person table has documents as follows:
// { "GovId": "TOYENC486FH", "FirstName" : "Brent" }
// { "GovId": "ROEE1C1AABH", "FirstName" : "Jim" }
// { "GovId": "YH844DA7LDB", "FirstName" : "Mary" }

string[] ids = {
    "TOYENC486FH",
    "ROEE1C1AABH",
    "YH844DA7LDB"
};
```

```
IAsyncResult<Person> result = await driver.Execute(async txn =>
{
    return await txn.Execute(txn.Query<Person>("SELECT * FROM Person WHERE GovId IN
 (?,?,?)", ids));
});

await foreach (Person person in result)
{
    Console.WriteLine(person.FirstName); // Prints Brent on first iteration.
    // Prints Jim on second iteration.
    // Prints Mary on third iteration.
}
```

L'exemple de code suivant utilise une liste C# comme valeur.

```
// Assumes that Person table has document as follows:
// { "GovId": "TOYENC486FH",
//   "FirstName" : "Brent",
//   "Vehicles": [
//      { "Make": "Volkswagen",
//        "Model": "Golf"},
//      { "Make": "Honda",
//        "Model": "Civic"}
//   ]
// }
// Person class is defined as follows:
// public class Person
// {
//     public string GovId { get; set; }
//     public string FirstName { get; set; }
//     public List<Vehicle> Vehicles { get; set; }
// }
// Vehicle class is defined as follows:
// public class Vehicle
// {
//     public string Make { get; set; }
//     public string Model { get; set; }
// }

List<Vehicle> vehicles = new List<Vehicle>
{
    new Vehicle
```

```
    {
        Make = "Volkswagen",
        Model = "Golf"
    },
    new Vehicle
    {
        Make = "Honda",
        Model = "Civic"
    }
};

IAsyncResult<Person> result = await driver.Execute(async txn =>
{
    return await txn.Execute(txn.Query<Person>("SELECT * FROM Person WHERE Vehicles
 = ?", vehicles));
});

await foreach (Person person in result)
{
    Console.WriteLine("{");
    Console.WriteLine($"  GovId: {person.GovId},");
    Console.WriteLine($"  FirstName: {person.FirstName},");
    Console.WriteLine("  Vehicles: [");
    foreach (Vehicle vehicle in person.Vehicles)
    {
        Console.WriteLine("  {");
        Console.WriteLine($"    Make: {vehicle.Make},");
        Console.WriteLine($"    Model: {vehicle.Model},");
        Console.WriteLine("  },");
    }
    Console.WriteLine("  ]");
    Console.WriteLine("}");
    // Prints:
    // {
    //     GovId: TOYENC486FH,
    //     FirstName: Brent,
    //     Vehicles: [
    //     {
    //         Make: Volkswagen,
    //         Model: Golf
    //     },
    //     {
    //         Make: Honda,
    //         Model: Civic
```

```
    //     },
    //     ]
    // }
}
```

Utilisation de la bibliothèque Ion

Async

```
// Assumes that Person table has documents as follows:
// { "GovId": "TOYENC486FH", "FirstName" : "Brent" }

IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("SELECT * FROM Person WHERE GovId = 'TOYENC486FH'");
});

await foreach (IIonValue row in result)
{
    Console.WriteLine(row.GetField("GovId").StringValue); // Prints TOYENC486FH.
    Console.WriteLine(row.GetField("FirstName").StringValue); // Prints Brent.
}
```

Sync

```
// Assumes that Person table has documents as follows:
// { "GovId": "TOYENC486FH", "FirstName" : "Brent" }

IResult result = driver.Execute(txn =>
{
    return txn.Execute("SELECT * FROM Person WHERE GovId = 'TOYENC486FH'");
});

foreach (IIonValue row in result)
{
    Console.WriteLine(row.GetField("GovId").StringValue); // Prints TOYENC486FH.
    Console.WriteLine(row.GetField("FirstName").StringValue); // Prints Brent.
}
```

> **Note**
>
> Lorsque vous exécutez une requête sans recherche indexée, elle appelle une analyse complète de la table. Dans cet exemple, nous recommandons de disposer d'un index sur le`GovId` terrain afin d'optimiser les performances. Si aucun index n'est activé`GovId`, les requêtes peuvent présenter une latence plus importante et peuvent également entraîner des exceptions de conflit OCC ou des délais de transaction.

L'exemple de code suivant utilise un paramètre de requête de type Ion.

Async

```
IValueFactory valueFactory = new ValueFactory();
IIonValue ionFirstName = valueFactory.NewString("Brent");

IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("SELECT * FROM Person WHERE FirstName = ?",
 ionFirstName);
});

await foreach (IIonValue row in result)
{
    Console.WriteLine(row.GetField("GovId").StringValue); // Prints TOYENC486FH.
    Console.WriteLine(row.GetField("FirstName").StringValue); // Prints Brent.
}
```

Sync

```
IValueFactory valueFactory = new ValueFactory();
IIonValue ionFirstName = valueFactory.NewString("Brent");

IResult result = driver.Execute(txn =>
{
    return txn.Execute("SELECT * FROM Person WHERE FirstName = ?", ionFirstName);
});

foreach (IIonValue row in result)
{
    Console.WriteLine(row.GetField("GovId").StringValue); // Prints TOYENC486FH.
```

```
    Console.WriteLine(row.GetField("FirstName").StringValue); // Prints Brent.
}
```

L'exemple de code suivant utilise plusieurs paramètres de requête.

Async

```
IIonValue ionGovId = valueFactory.NewString("TOYENC486FH");
IIonValue ionFirstName = valueFactory.NewString("Brent");

IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("SELECT * FROM Person WHERE GovId = ? AND FirstName
 = ?", ionGovId, ionFirstName);
});

await foreach (IIonValue row in result)
{
    Console.WriteLine(row.GetField("GovId").StringValue); // Prints TOYENC486FH.
    Console.WriteLine(row.GetField("FirstName").StringValue); // Prints Brent.
}
```

Sync

```
IIonValue ionGovId = valueFactory.NewString("TOYENC486FH");
IIonValue ionFirstName = valueFactory.NewString("Brent");

IResult result = driver.Execute(txn =>
{
    return txn.Execute("SELECT * FROM Person WHERE GovId = ? AND FirstName = ?",
 ionGovId, ionFirstName);
});

foreach (IIonValue row in result)
{
    Console.WriteLine(row.GetField("GovId").StringValue); // Prints TOYENC486FH.
    Console.WriteLine(row.GetField("FirstName").StringValue); // Prints Brent.
}
```

L'exemple de code suivant utilise une liste de paramètres de requête.

Async

```
// Assumes that Person table has documents as follows:
// { "GovId": "TOYENC486FH", "FirstName" : "Brent" }
// { "GovId": "ROEE1C1AABH", "FirstName" : "Jim" }
// { "GovId": "YH844DA7LDB", "FirstName" : "Mary" }

IIonValue[] ionIds = {
    valueFactory.NewString("TOYENC486FH"),
    valueFactory.NewString("ROEE1C1AABH"),
    valueFactory.NewString("YH844DA7LDB")
};

IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("SELECT * FROM Person WHERE GovId IN (?,?,?)", ionIds);
});

await foreach (IIonValue row in result)
{
    Console.WriteLine(row.GetField("FirstName").StringValue); // Prints Brent on
 first iteration.
                                                              // Prints Jim on
 second iteration.
                                                              // Prints Mary on
 third iteration.
}
```

Sync

```
// Assumes that Person table has documents as follows:
// { "GovId": "TOYENC486FH", "FirstName" : "Brent" }
// { "GovId": "ROEE1C1AABH", "FirstName" : "Jim" }
// { "GovId": "YH844DA7LDB", "FirstName" : "Mary" }

IIonValue[] ionIds = {
    valueFactory.NewString("TOYENC486FH"),
    valueFactory.NewString("ROEE1C1AABH"),
    valueFactory.NewString("YH844DA7LDB")
};

IResult result = driver.Execute(txn =>
{
```

```
    return txn.Execute("SELECT * FROM Person WHERE GovId IN (?,?,?)", ionIds);
});

foreach (IIonValue row in result)
{
    Console.WriteLine(row.GetField("FirstName").StringValue); // Prints Brent on
 first iteration.
                                                              // Prints Jim on
 second iteration.
                                                              // Prints Mary on
 third iteration.
}
```

L'exemple de code suivant utilise une liste d'ions comme valeur. Pour plus d'informations sur l'utilisation des différents types d'ions, consultezUtilisation des types de données Amazon Ion dans Amazon QLDB.

Async

```
// Assumes that Person table has document as follows:
// { "GovId": "TOYENC486FH",
//   "FirstName" : "Brent",
//   "Vehicles": [
//      { "Make": "Volkswagen",
//        "Model": "Golf"},
//      { "Make": "Honda",
//        "Model": "Civic"}
//   ]
// }

IIonValue ionVehicle1 = valueFactory.NewEmptyStruct();
ionVehicle1.SetField("Make", valueFactory.NewString("Volkswagen"));
ionVehicle1.SetField("Model", valueFactory.NewString("Golf"));

IIonValue ionVehicle2 = valueFactory.NewEmptyStruct();
ionVehicle2.SetField("Make", valueFactory.NewString("Honda"));
ionVehicle2.SetField("Model", valueFactory.NewString("Civic"));

IIonValue ionVehicles =  valueFactory.NewEmptyList();
ionVehicles.Add(ionVehicle1);
ionVehicles.Add(ionVehicle2);
```

```
IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("SELECT * FROM Person WHERE Vehicles = ?",
 ionVehicles);
});

await foreach (IIonValue row in result)
{
    Console.WriteLine(row.ToPrettyString());
    // Prints:
    // {
    //     GovId: "TOYENC486FN",
    //     FirstName: "Brent",
    //     Vehicles: [
    //     {
    //         Make: "Volkswagen",
    //         Model: "Golf"
    //     },
    //     {
    //         Make: "Honda",
    //         Model: "Civic"
    //     }
    //     ]
    // }
}
```

Sync

```
// Assumes that Person table has document as follows:
// { "GovId": "TOYENC486FH",
//   "FirstName" : "Brent",
//   "Vehicles": [
//      { "Make": "Volkswagen",
//        "Model": "Golf"},
//      { "Make": "Honda",
//        "Model": "Civic"}
//   ]
// }

IIonValue ionVehicle1 = valueFactory.NewEmptyStruct();
ionVehicle1.SetField("Make", valueFactory.NewString("Volkswagen"));
ionVehicle1.SetField("Model", valueFactory.NewString("Golf"));
```

```
IIonValue ionVehicle2 = valueFactory.NewEmptyStruct();
ionVehicle2.SetField("Make", valueFactory.NewString("Honda"));
ionVehicle2.SetField("Model", valueFactory.NewString("Civic"));

IIonValue ionVehicles =  valueFactory.NewEmptyList();
ionVehicles.Add(ionVehicle1);
ionVehicles.Add(ionVehicle2);

IResult result = driver.Execute(txn =>
{
    return txn.Execute("SELECT * FROM Person WHERE Vehicles = ?", ionVehicles);
});

foreach (IIonValue row in result)
{
    Console.WriteLine(row.ToPrettyString());
    // Prints:
    // {
    //     GovId: "TOYENC486FN",
    //     FirstName: "Brent",
    //     Vehicles: [
    //     {
    //         Make: "Volkswagen",
    //         Model: "Golf"
    //     },
    //     {
    //         Make: "Honda",
    //         Model: "Civic"
    //     }
    //     ]
    // }
}
```

Insertion de documents

L'exemple de code suivant insère des types de données Ion.

```
string govId = "TOYENC486FH";

Person person = new Person
{
    GovId = "TOYENC486FH",
    FirstName = "Brent"
```

```
};

await driver.Execute(async txn =>
{
    // Check if a document with GovId:TOYENC486FH exists
    // This is critical to make this transaction idempotent
    IAsyncResult<Person> result = await txn.Execute(txn.Query<Person>("SELECT * FROM
 Person WHERE GovId = ?", govId));

    // Check if there is a record in the cursor.
    int count = await result.CountAsync();
    if (count > 0)
    {
        // Document already exists, no need to insert
        return;
    }

    // Insert the document.
    await txn.Execute(txn.Query<Document>("INSERT INTO Person ?", person));
});
```

Utilisation de la bibliothèque Ion

Async

```
IIonValue ionGovId = valueFactory.NewString("TOYENC486FH");

IIonValue ionPerson = valueFactory.NewEmptyStruct();
ionPerson.SetField("GovId", valueFactory.NewString("TOYENC486FH"));
ionPerson.SetField("FirstName", valueFactory.NewString("Brent"));

await driver.Execute(async txn =>
{
    // Check if a document with GovId:TOYENC486FH exists
    // This is critical to make this transaction idempotent
    IAsyncResult result = await txn.Execute("SELECT * FROM Person WHERE GovId = ?",
 ionGovId);

    // Check if there is a record in the cursor.
    int count = await result.CountAsync();
    if (count > 0)
    {
        // Document already exists, no need to insert
```

```
        return;
    }

    // Insert the document.
    await txn.Execute("INSERT INTO Person ?", ionPerson);
});
```

Sync

```
IIonValue ionGovId = valueFactory.NewString("TOYENC486FH");

IIonValue ionPerson = valueFactory.NewEmptyStruct();
ionPerson.SetField("GovId", valueFactory.NewString("TOYENC486FH"));
ionPerson.SetField("FirstName", valueFactory.NewString("Brent"));

driver.Execute(txn =>
{
    // Check if a document with GovId:TOYENC486FH exists
    // This is critical to make this transaction idempotent
    IResult result = txn.Execute("SELECT * FROM Person WHERE GovId = ?", ionGovId);

    // Check if there is a record in the cursor.
    int count = result.Count();
    if (count > 0)
    {
        // Document already exists, no need to insert
        return;
    }

    // Insert the document.
    txn.Execute("INSERT INTO Person ?", ionPerson);
});
```

Cette transaction insère un document dans le`Person` tableau. Avant de l'insérer, il vérifie d'abord si le document existe déjà dans le tableau. Ce contrôle confère à la transaction un caractère idempotent. Même si vous exécutez cette transaction plusieurs fois, elle n'aura aucun effet secondaire imprévu.

> **Note**
>
> Dans cet exemple, nous recommandons de disposer d'un index sur le`GovId` terrain afin d'optimiser les performances. Si aucun index n'est activé`GovId`, les instructions peuvent présenter une latence plus importante et peuvent également entraîner des exceptions de conflit OCC ou des délais de transaction.

Insertion de plusieurs documents dans une seule déclaration

Pour insérer plusieurs documents à l'aide d'une seuleINSERT instruction, vous pouvez transmettre un`List` paramètre C# à l'instruction comme suit.

```
Person person1 = new Person
{
    FirstName = "Brent",
    GovId = "TOYENC486FH"
};

Person person2 = new Person
{
    FirstName = "Jim",
    GovId = "ROEE1C1AABH"
};

List<Person> people = new List<Person>();
people.Add(person1);
people.Add(person2);

IAsyncResult<Document> result = await driver.Execute(async txn =>
{
    return await txn.Execute(txn.Query<Document>("INSERT INTO Person ?", people));
});

await foreach (Document row in result)
{
    Console.WriteLine("{ documentId: " + row.DocumentId + " }");
    // The statement returns the created documents' ID:
    // { documentId: 6BFt5eJQDFLBW2aR8LPw42 }
    // { documentId: K5Zrcb6N3gmIEHgGhwoyKF }
}
```

Utilisation de la bibliothèque Ion

Pour insérer plusieurs documents à l'aide d'une seuleINSERT instruction, vous pouvez transmettre un paramètre de type Ion list à l'instruction comme suit.

Async

```
IIonValue ionPerson1 = valueFactory.NewEmptyStruct();
ionPerson1.SetField("FirstName", valueFactory.NewString("Brent"));
ionPerson1.SetField("GovId", valueFactory.NewString("TOYENC486FH"));

IIonValue ionPerson2 = valueFactory.NewEmptyStruct();
ionPerson2.SetField("FirstName", valueFactory.NewString("Jim"));
ionPerson2.SetField("GovId", valueFactory.NewString("ROEE1C1AABH"));

IIonValue ionPeople = valueFactory.NewEmptyList();
ionPeople.Add(ionPerson1);
ionPeople.Add(ionPerson2);

IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("INSERT INTO Person ?", ionPeople);
});

await foreach (IIonValue row in result)
{
    Console.WriteLine(row.ToPrettyString());
    // The statement returns the created documents' ID:
    // {
    //     documentId: "6BFt5eJQDFLBW2aR8LPw42"
    // }
    //
    // {
    //     documentId: "K5Zrcb6N3gmIEHgGhwoyKF"
    // }
}
```

Sync

```
IIonValue ionPerson1 = valueFactory.NewEmptyStruct();
ionPerson1.SetField("FirstName", valueFactory.NewString("Brent"));
ionPerson1.SetField("GovId", valueFactory.NewString("TOYENC486FH"));
```

```
IIonValue ionPerson2 = valueFactory.NewEmptyStruct();
ionPerson2.SetField("FirstName", valueFactory.NewString("Jim"));
ionPerson2.SetField("GovId", valueFactory.NewString("ROEE1C1AABH"));

IIonValue ionPeople = valueFactory.NewEmptyList();
ionPeople.Add(ionPerson1);
ionPeople.Add(ionPerson2);

IResult result = driver.Execute(txn =>
{
    return txn.Execute("INSERT INTO Person ?", ionPeople);
});

foreach (IIonValue row in result)
{
    Console.WriteLine(row.ToPrettyString());
    // The statement returns the created documents' ID:
    // {
    //     documentId: "6BFt5eJQDFLBW2aR8LPw42"
    // }
    //
    // {
    //     documentId: "K5Zrcb6N3gmIEHgGhwoyKF"
    // }
}
```

Vous ne placez pas l'espace réservé à la variable (?) entre crochets (<<...>>) lorsque vous transmettez une liste Ion. Dans les instructions manuelles de PartiQL, les crochets doubles indiquent une collection non ordonnée appelée sac.

Mise à jour des documents

```
string govId = "TOYENC486FH";
string firstName = "John";

IAsyncResult<Document> result = await driver.Execute(async txn =>
{
    return await txn.Execute(txn.Query<Document>("UPDATE Person SET FirstName = ? WHERE
 GovId = ?", firstName , govId));
});

await foreach (Document row in result)
```

```
{
    Console.WriteLine("{ documentId: " + row.DocumentId + " }");
    // The statement returns the updated document ID:
    // { documentId: Djg30Zoltqy5M4BFsA2jSJ }
}
```

Utilisation de la bibliothèque Ion

Async

```
IIonValue ionGovId = valueFactory.NewString("TOYENC486FH");
IIonValue ionFirstName = valueFactory.NewString("John");

IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("UPDATE Person SET FirstName = ? WHERE GovId = ?",
 ionFirstName , ionGovId);
});

await foreach (IIonValue row in result)
{
    Console.WriteLine(row.ToPrettyString());
    // The statement returns the updated document ID:
    // {
    //     documentId: "Djg30Zoltqy5M4BFsA2jSJ"
    // }
}
```

Sync

```
IIonValue ionGovId = valueFactory.NewString("TOYENC486FH");
IIonValue ionFirstName = valueFactory.NewString("John");

IResult result = driver.Execute(txn =>
{
    return txn.Execute("UPDATE Person SET FirstName = ? WHERE GovId = ?",
 ionFirstName , ionGovId);
});

foreach (IIonValue row in result)
{
    Console.WriteLine(row.ToPrettyString());
    // The statement returns the updated document ID:
```

```
    // {
    //     documentId: "Djg30Zoltqy5M4BFsA2jSJ"
    // }
}
```

> **Note**
>
> Dans cet exemple, nous recommandons de disposer d'un index sur le`GovId` terrain afin d'optimiser les performances. Si aucun index n'est activé`GovId`, les instructions peuvent présenter une latence plus importante et peuvent également entraîner des exceptions de conflit OCC ou des délais de transaction.

Supprimer des documents

```
string govId = "TOYENC486FH";

IAsyncResult<Document> result = await driver.Execute(async txn =>
{
    return await txn.Execute(txn.Query<Document>("DELETE FROM Person WHERE GovId = ?",
 govId));
});

await foreach (Document row in result)
{
    Console.WriteLine("{ documentId: " + row.DocumentId + " }");
    // The statement returns the updated document ID:
    // { documentId: Djg30Zoltqy5M4BFsA2jSJ }
}
```

Utilisation de la bibliothèque Ion

Async

```
IIonValue ionGovId = valueFactory.NewString("TOYENC486FH");

IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("DELETE FROM Person WHERE GovId = ?", ionGovId);
});
```

```
await foreach (IIonValue row in result)
{
    Console.WriteLine(row.ToPrettyString());
    // The statement returns the deleted document ID:
    // {
    //     documentId: "Djg30Zoltqy5M4BFsA2jSJ"
    // }
}
```

Sync

```
IIonValue ionGovId = valueFactory.NewString("TOYENC486FH");

IResult result = driver.Execute(txn =>
{
    return txn.Execute("DELETE FROM Person WHERE GovId = ?", ionGovId);
});

foreach (IIonValue row in result)
{
    Console.WriteLine(row.ToPrettyString());
    // The statement returns the deleted document ID:
    // {
    //     documentId: "Djg30Zoltqy5M4BFsA2jSJ"
    // }
}
```

> **Note**
>
> Dans cet exemple, nous recommandons de disposer d'un index sur le`GovId` terrain afin d'optimiser les performances. Si aucun index n'est activé`GovId`, les instructions peuvent présenter une latence plus importante et peuvent également entraîner des exceptions de conflit OCC ou des délais de transaction.

Exécution de plusieurs relevés dans une transaction

```
// This code snippet is intentionally trivial. In reality you wouldn't do this because
 you'd
```

```
// set your UPDATE to filter on vin and insured, and check if you updated something or
 not.
public static async Task<bool> InsureVehicle(IAsyncQldbDriver driver, string vin)
{
    return await driver.Execute(async txn =>
    {
        // Check if the vehicle is insured.
        Amazon.QLDB.Driver.Generic.IAsyncResult<Vehicle> result = await txn.Execute(
            txn.Query<Vehicle>("SELECT insured FROM Vehicles WHERE vin = ? AND insured
 = FALSE", vin));

        if (await result.CountAsync() > 0)
        {
            // If the vehicle is not insured, insure it.
            await txn.Execute(
                txn.Query<Document>("UPDATE Vehicles SET insured = TRUE WHERE vin = ?",
 vin));
            return true;
        }
        return false;
    });
}
```

Utilisation de la bibliothèque Ion

Async

```
// This code snippet is intentionally trivial. In reality you wouldn't do this
 because you'd
// set your UPDATE to filter on vin and insured, and check if you updated something
 or not.
public static async Task<bool> InsureVehicle(IAsyncQldbDriver driver, string vin)
{
    ValueFactory valueFactory = new ValueFactory();
    IIonValue ionVin = valueFactory.NewString(vin);

    return await driver.Execute(async txn =>
    {
        // Check if the vehicle is insured.
        Amazon.QLDB.Driver.IAsyncResult result = await txn.Execute(
            "SELECT insured FROM Vehicles WHERE vin = ? AND insured = FALSE",
 ionVin);
```

```
        if (await result.CountAsync() > 0)
        {
            // If the vehicle is not insured, insure it.
            await txn.Execute(
                "UPDATE Vehicles SET insured = TRUE WHERE vin = ?", ionVin);
            return true;
        }
        return false;
    });
}
```

Nouvelle tentative

Pour plus d'informations sur la logique de nouvelle tentative intégrée au pilote, reportez-vous à la sectionComprendre la stratégie de nouvelle tentative avec le pilote dans Amazon QLDB.

Mise en œuvre des contraintes d'unicité

QLDB ne prend pas en charge les index uniques, mais vous pouvez implémenter ce comportement dans votre application.

Supposons que vous souhaitiez implémenter une contrainte d'unicité sur le`GovId` champ de la`Person` table. Pour ce faire, vous pouvez écrire une transaction qui effectue les actions suivantes :

1. Affirmez que la table ne contient aucun document existant portant une valeur spécifiée`GovId`.
2. Insérez le document si l'assertion est acceptée.

Si une transaction concurrente passe simultanément l'assertion, seule l'une des transactions sera validée avec succès. L'autre transaction échouera avec une exception de conflit OCC.

L'exemple de code suivant montre comment implémenter cette logique de contrainte d'unicité.

```
string govId = "TOYENC486FH";

Person person = new Person
{
    GovId = "TOYENC486FH",
    FirstName = "Brent"
};

await driver.Execute(async txn =>
```

```
{
    // Check if a document with GovId:TOYENC486FH exists
    // This is critical to make this transaction idempotent
    IAsyncResult<Person> result = await txn.Execute(txn.Query<Person>("SELECT * FROM
 Person WHERE GovId = ?", govId));

    // Check if there is a record in the cursor.
    int count = await result.CountAsync();
    if (count > 0)
    {
        // Document already exists, no need to insert
        return;
    }

    // Insert the document.
    await txn.Execute(txn.Query<Document>("INSERT INTO Person ?", person));
});
```

Utilisation de la bibliothèque Ion

Async

```
IIonValue ionGovId = valueFactory.NewString("TOYENC486FH");

IIonValue ionPerson = valueFactory.NewEmptyStruct();
ionPerson.SetField("GovId", valueFactory.NewString("TOYENC486FH"));
ionPerson.SetField("FirstName", valueFactory.NewString("Brent"));

await driver.Execute(async txn =>
{
    // Check if a document with GovId:TOYENC486FH exists
    // This is critical to make this transaction idempotent
    IAsyncResult result = await txn.Execute("SELECT * FROM Person WHERE GovId = ?",
 ionGovId);

    // Check if there is a record in the cursor.
    int count = await result.CountAsync();
    if (count > 0)
    {
        // Document already exists, no need to insert
        return;
    }
```

```
    // Insert the document.
    await txn.Execute("INSERT INTO Person ?", ionPerson);
});
```

Sync

```
IIonValue ionGovId = valueFactory.NewString("TOYENC486FH");

IIonValue ionPerson = valueFactory.NewEmptyStruct();
ionPerson.SetField("GovId", valueFactory.NewString("TOYENC486FH"));
ionPerson.SetField("FirstName", valueFactory.NewString("Brent"));

driver.Execute(txn =>
{
    // Check if a document with GovId:TOYENC486FH exists
    // This is critical to make this transaction idempotent
    IResult result = txn.Execute("SELECT * FROM Person WHERE GovId = ?", ionGovId);

    // Check if there is a record in the cursor.
    int count = result.Count();
    if (count > 0)
    {
        // Document already exists, no need to insert
        return;
    }

    // Insert the document.
    txn.Execute("INSERT INTO Person ?", ionPerson);
});
```

> **Note**
>
> Dans cet exemple, nous recommandons de disposer d'un index sur le`GovId` terrain afin d'optimiser les performances. Si aucun index n'est activé`GovId`, les instructions peuvent présenter une latence plus importante et peuvent également entraîner des exceptions de conflit OCC ou des délais de transaction.

Utilisation d'Amazon Ion

Vous disposez de plusieurs options pour traiter les données Amazon Ion dans QLDB de différentes options. Vous pouvez utiliser la bibliothèque d'ions pour créer et modifier des valeurs d'ions. Vous pouvez également utiliser le mappeur d'objets Ion pour mapper des objets CLR simples (POCO) en C# vers et depuis des valeurs Ion. La version 1.3.0 du pilote QLDB pour .NET introduit la prise en charge du mappeur d'objets Ion.

Les sections suivantes fournissent des exemples de code de traitement de données ioniques utilisant les deux techniques.

Table des matières

Importation du module Ion

```
using Amazon.IonObjectMapper;
```

Utilisation de la bibliothèque Ion

```
using Amazon.IonDotnet.Builders;
```

Création de types Ion

L'exemple de code suivant montre comment créer des valeurs Ion à partir d'objets C# à l'aide de l'ion Object Mapper.

```
// Assumes that Person class is defined as follows:
// public class Person
// {
//     public string FirstName { get; set; }
//     public int Age { get; set; }
// }

// Initialize the Ion Object Mapper
```

```
IonSerializer ionSerializer = new IonSerializer();

// The C# object to be serialized
Person person = new Person
{
    FirstName = "John",
    Age = 13
};

// Serialize the C# object into stream using the Ion Object Mapper
Stream stream = ionSerializer.Serialize(person);

// Load will take in stream and return a datagram; a top level container of Ion values.
IIonValue ionDatagram = IonLoader.Default.Load(stream);

// To get the Ion value within the datagram, we call GetElementAt(0).
IIonValue ionPerson = ionDatagram.GetElementAt(0);

Console.WriteLine(ionPerson.GetField("firstName").StringValue);
Console.WriteLine(ionPerson.GetField("age").IntValue);
```

Utilisation de la bibliothèque Ion

Les exemples de code suivants montrent les deux manières de créer des valeurs Ion à l'aide de la bibliothèque Ion.

Utilisation de **`ValueFactory`**

```
using Amazon.IonDotnet.Tree;
using Amazon.IonDotnet.Tree.Impl;

IValueFactory valueFactory = new ValueFactory();

IIonValue ionPerson = valueFactory.NewEmptyStruct();
ionPerson.SetField("firstName", valueFactory.NewString("John"));
ionPerson.SetField("age", valueFactory.NewInt(13));

Console.WriteLine(ionPerson.GetField("firstName").StringValue);
Console.WriteLine(ionPerson.GetField("age").IntValue);
```

Utilisation de **`IonLoader`**

```
using Amazon.IonDotnet.Builders;
```

```
using Amazon.IonDotnet.Tree;

// Load will take in Ion text and return a datagram; a top level container of Ion
 values.
IIonValue ionDatagram = IonLoader.Default.Load("{firstName: \"John\", age: 13}");

// To get the Ion value within the datagram, we call GetElementAt(0).
IIonValue ionPerson = ionDatagram.GetElementAt(0);

Console.WriteLine(ionPerson.GetField("firstName").StringValue);
Console.WriteLine(ionPerson.GetField("age").IntValue);
```

Obtenir un dump binaire Ion

```
// Initialize the Ion Object Mapper with Ion binary serialization format
IonSerializer ionSerializer = new IonSerializer(new IonSerializationOptions
{
    Format = IonSerializationFormat.BINARY
});

// The C# object to be serialized
Person person = new Person
{
    FirstName = "John",
    Age = 13
};

MemoryStream stream = (MemoryStream) ionSerializer.Serialize(person);
Console.WriteLine(BitConverter.ToString(stream.ToArray()));
```

Utilisation de la bibliothèque Ion

```
// ionObject is an Ion struct
MemoryStream stream = new MemoryStream();
using (var writer = IonBinaryWriterBuilder.Build(stream))
{
    ionObject.WriteTo(writer);
    writer.Finish();
}

Console.WriteLine(BitConverter.ToString(stream.ToArray()));
```

Obtenir un dump de texte Ion

```
// Initialize the Ion Object Mapper
IonSerializer ionSerializer = new IonSerializer(new IonSerializationOptions
{
    Format = IonSerializationFormat.TEXT
});

// The C# object to be serialized
Person person = new Person
{
    FirstName = "John",
    Age = 13
};

MemoryStream stream = (MemoryStream) ionSerializer.Serialize(person);
Console.WriteLine(System.Text.Encoding.UTF8.GetString(stream.ToArray()));
```

Utilisation de la bibliothèque Ion

```
// ionObject is an Ion struct
StringWriter sw = new StringWriter();
using (var writer = IonTextWriterBuilder.Build(sw))
{
    ionObject.WriteTo(writer);
    writer.Finish();
}

Console.WriteLine(sw.ToString());
```

Pour plus d'informations sur l'utilisation d'Ion, consultez la documentation Amazon Ion sur GitHub. Pour plus d'exemples de code illustrant l'utilisation d'Ion dans QLDB, consultezUtilisation des types de données Amazon Ion dans Amazon QLDB.

Pilote Amazon QLDB pour Go

Pour utiliser les données de votre registre, vous pouvez vous connecter à Amazon QLDB depuis votre application Go à l'aide d'un piloteAWS fourni. Les rubriques suivantes décrivent comment installer le pilote QLDB pour Go.

Rubriques

Ressources pour les conducteurs

Pour plus d'informations sur les fonctionnalités prises en charge par le pilote Go, consultez les ressources suivantes :

- Référence de l'API : 3.x, 2.x, 1.x
- Code source du pilote (GitHub)
- Livre de recettes Amazon Ion

Prérequis

Avant de démarrer le pilote QLDB pour Go, vous devez effectuer les opérations suivantes :

1. Suivez les instructions deAWS configuration dansAccès à Amazon QLDB. Cela inclut les éléments suivants :

    1. Inscrivez-vous àAWS.
    2. Créez un utilisateur avec les autorisations QLDB appropriées.
    3. Accorder un accès par programmation.
2. (Facultatif) Installez l'environnement de développement intégré (IDE) de votre choix. Pour obtenir la liste des IDE couramment utilisés pour Go, consultez la section Plug-ins et IDE de l'éditeur sur le site Web de Go.
3. Téléchargez et installez l'une des versions suivantes de Go à partir du site de téléchargement de Go :

    - 1.15 ou version ultérieure — pilote QLDB pour Go v3
    - 1.14 — Pilote QLDB pour Go v1 ou v2
4. Configurez votre environnement de développement pour AWS SDK for Go:

1. Configurez vosAWS informations d'identification. Nous vous recommandons de créer un fichier d'informations d'identification partagées.

   Pour obtenir des instructions, consultez la section Spécification des informations d'identification dans le Guide duAWS SDK for Go développeur.

2. Définissez votre valeur par défautRégion AWS. Pour savoir comment procéder, voir Spécifier leRégion AWS.

   Pour obtenir la liste complète des régions disponibles, consultez les points de terminaison et les quotas Amazon QLDB dans le Références générales AWS.

Vous pouvez ensuite configurer un exemple d'application de base et exécuter des exemples de code abrégé, ou vous pouvez installer le pilote dans un projet Go existant.

- Pour installer le pilote QLDB et leAWS SDK for Go dans un projet existant, passez àInstallation.
- Pour configurer un projet et exécuter des exemples de code abrégé illustrant les transactions de données de base sur un registre, consultez leDidacticiel de démarrage rapide.

Installation

Le pilote QLDB pour Go est open source dans le GitHub référentiel awslabs/amazon-qldb-driver-go. QLDB prend en charge les versions de pilotes suivantes et leurs dépendances à Go.

Versions du pilote	Version Go	État	Dernière date de parution
1.x	1.14 ou version ultérieure	Version de production	16 juin 2021
2.x	1.14 ou version ultérieure	Version de production	21 juillet 2021
3.x	1.15 ou version ultérieure	Version de production	10 novembre 2022

Pour installer le pilote

1. Assurez-vous que votre projet utilise des modules Go pour installer les dépendances du projet.
2. Dans le répertoire de votre projet, entrez la `go get` commande suivante.

   3.x

   ```
   $ go get -u github.com/awslabs/amazon-qldb-driver-go/v3/qldbdriver
   ```

   2.x

   ```
   $ go get -u github.com/awslabs/amazon-qldb-driver-go/v2/qldbdriver
   ```

L'installation du pilote installe également ses dépendances, notamment les packages AWS SDK for Goor AWS SDK for Gov2 et Amazon Ion.

Pour des exemples de code courts expliquant comment exécuter des transactions de données de base sur un registre, consultez leRéférence de livre de recettes.

Pilote Amazon QLDB pour Go — Tutoriel de démarrage rapide

Dans ce tutoriel, vous apprendrez comment configurer une application simple à l'aide de la dernière version du pilote Amazon QLDB pour Go. Ce guide inclut les étapes d'installation du pilote et des exemples de code court des opérations de base de création, lecture, mise à jour et suppression (CRUD).

Rubriques

- Étape 8 : Rechercher le document mis à jour
- Étape 9 : Suppression de la table
- Exécution de l'application complète

Prérequis

Avant de commencer, veillez à exécuter les actions suivantes :

1. Complétez le pilotePrérequis for the Go, si vous ne l'avez déjà fait. Cela inclut l'inscriptionAWS, l'octroi d'un accès programmatique pour le développement et l'installation de Go.
2. Créez un registre nomméquick-start.

   Pour savoir comment créer un registre, consultezOpérations de base pour les registres Amazon QLDB ouÉtape 1 : Créer un nouveau registre dans Prise en main de la console.

Étape 1 : Installation du pilote

Assurez-vous que votre projet utilise des modules Go pour installer les dépendances du projet.

Dans le répertoire de votre projet, entrez lago get commande suivante.

```
$ go get -u github.com/awslabs/amazon-qldb-driver-go/v3/qldbdriver
```

L'installation du pilote installe également ses dépendances, notamment les packages AWS SDK for Gov2 et Amazon Ion.

Étape 2 : Importer les packages

Importez lesAWS packages suivants.

```
import (
    "context"
    "fmt"

    "github.com/amzn/ion-go/ion"
    "github.com/aws/aws-sdk-go/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/qldbSession"
    "github.com/awslabs/amazon-qldb-driver-go/v3/qldbdriver"
```

```
)
```

Étape 3 : Initialiser le pilote

Initialisez une instance du pilote qui se connecte au registre nomméquick-start.

```
cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
  panic(err)
}

qldbSession := qldbsession.NewFromConfig(cfg, func(options *qldbsession.Options) {
    options.Region = "us-east-1"
})
driver, err := qldbdriver.New(
    "quick-start",
    qldbSession,
    func(options *qldbdriver.DriverOptions) {
        options.LoggerVerbosity = qldbdriver.LogInfo
    })
if err != nil {
    panic(err)
}

defer driver.Shutdown(context.Background())
```

> **Note**
>
> Dans cet exemple de code, remplacez *us-east-1* par l'Région AWSendroit où vous avez créé votre registre.

Étape 4 : Création d'une table et d'un index

L'exemple de code suivant montre comment exécuterCREATE TABLE lesCREATE INDEX instructions.

```
_, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
    _, err := txn.Execute("CREATE TABLE People")
    if err != nil {
```

```
        return nil, err
    }

    // When working with QLDB, it's recommended to create an index on fields we're
 filtering on.
    // This reduces the chance of OCC conflict exceptions with large datasets.
    _, err = txn.Execute("CREATE INDEX ON People (firstName)")
    if err != nil {
        return nil, err
    }

    _, err = txn.Execute("CREATE INDEX ON People (age)")
    if err != nil {
        return nil, err
    }

    return nil, nil
})
if err != nil {
    panic(err)
}
```

Ce code crée une table nomméePeople et indexe lesage champsfirstName et de cette table. Les index sont nécessaires pour optimiser les performances des requêtes et contribuer à limiter les exceptions de conflit liées au contrôle optimiste de la concurrence (OCC).

Étape 5 : Insertion d'un document

Les exemples de code suivants montrent comment exécuter uneINSERT instruction. QLDB prend en charge le langage de requête PartiQL (compatible SQL) et le format de données Amazon Ion (surensemble de JSON).

Utilisation de PartiQL littéral

Le code suivant insère un document dans laPeople table à l'aide d'une instruction PartiQL littérale de chaîne.

```
_, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
    return txn.Execute("INSERT INTO People {'firstName': 'Jane', 'lastName': 'Doe',
 'age': 77}")
})
if err != nil {
```

```
    panic(err)
}
```

Utilisation des types de données Ion

À l'instar du package JSON intégré de Go, vous pouvez regrouper et dégrouper les types de données Go vers et depuis Ion.

1. Supposons que vous avez la structure Go suivante nomméePerson.

```
type Person struct {
    FirstName string `ion:"firstName"`
    LastName  string `ion:"lastName"`
    Age       int    `ion:"age"`
}
```

2. Créez une instance de Person.

```
person := Person{"John", "Doe", 54}
```

Le pilote gèrepersonpour vous une représentation textuelle codée par ions de.

> **Important**
>
> Pour que Marshal et Unmarshal fonctionnent correctement, les noms des champs de la structure de données Go doivent être exportés (première lettre en majuscule).

3. Transmettez l'personinstance à laExecute méthode de la transaction.

```
_, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
    return txn.Execute("INSERT INTO People ?", person)
})
if err != nil {
    panic(err)
}
```

Cet exemple utilise un point d'interrogation (?) comme espace réservé variable pour transmettre les informations du document à l'instruction. Lorsque vous utilisez des espaces réservés, vous devez transmettre une valeur de texte codée par ions.

> **Tip**
>
> Pour insérer plusieurs documents à l'aide d'une seuleINSERT instruction, vous pouvez transmettre un paramètre de type list à l'instruction comme suit.
>
> ```
> // people is a list
> txn.Execute("INSERT INTO People ?", people)
> ```
>
> Vous ne placez pas l'espace réservé à la variable (?) entre crochets (<<...>>) lorsque vous transmettez une liste. Dans les instructions manuelles de PartiQL, les crochets doubles indiquent une collection non ordonnée appelée sac.

Étape 6 : Interroger le document

L'exemple de code suivant montre comment exécuter uneSELECT instruction.

```
p, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
    result, err := txn.Execute("SELECT firstName, lastName, age FROM People WHERE age =
 54")
    if err != nil {
        return nil, err
    }

    // Assume the result is not empty
    hasNext := result.Next(txn)
    if !hasNext && result.Err() != nil {
        return nil, result.Err()
    }

    ionBinary := result.GetCurrentData()

    temp := new(Person)
    err = ion.Unmarshal(ionBinary, temp)
    if err != nil {
        return nil, err
    }

    return *temp, nil
```

```
})
if err != nil {
    panic(err)
}

var returnedPerson Person
returnedPerson = p.(Person)

if returnedPerson != person {
    fmt.Print("Queried result does not match inserted struct")
}
```

Cet exemple interroge votre document à partir du`People` tableau, suppose que le jeu de résultats n'est pas vide et renvoie votre document à partir du résultat.

Étape 7 : mise à jour du document

L'exemple de code suivant montre comment exécuter une`UPDATE` instruction.

```
person.Age += 10

_, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
    return txn.Execute("UPDATE People SET age = ? WHERE firstName = ?", person.Age,
 person.FirstName)
})
if err != nil {
    panic(err)
}
```

Étape 8 : Rechercher le document mis à jour

L'exemple de code suivant interroge la`People` table par`firstName` et renvoie tous les documents du jeu de résultats.

```
p, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
    result, err := txn.Execute("SELECT firstName, lastName, age FROM People WHERE
 firstName = ?", person.FirstName)
    if err != nil {
        return nil, err
```

```
    }

    var people []Person
    for result.Next(txn) {
        ionBinary := result.GetCurrentData()

        temp := new(Person)
        err = ion.Unmarshal(ionBinary, temp)
        if err != nil {
            return nil, err
        }

        people = append(people, *temp)
    }
    if result.Err() != nil {
        return nil, result.Err()
    }

    return people, nil
})
if err != nil {
    panic(err)
}

var people []Person
people = p.([]Person)

updatedPerson := Person{"John", "Doe", 64}
if people[0] != updatedPerson {
    fmt.Print("Queried result does not match updated struct")
}
```

Étape 9 : Suppression de la table

L'exemple de code suivant montre comment exécuter une`DROP TABLE` instruction.

```
_, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
    return txn.Execute("DROP TABLE People")
})
if err != nil {
    panic(err)
}
```

Exécution de l'application complète

L'exemple de code suivant représente la version complète de l'application. Au lieu de suivre les étapes précédentes individuellement, vous pouvez également copier et exécuter cet exemple de code du début à la fin. Cette application montre certaines opérations CRUD de base sur le registre nomméquick-start.

> **Note**
>
> Avant d'exécuter ce code, assurez-vous qu'aucune table active n'est déjà nomméePeople dans lequick-start registre.

```
package main

import (
    "context"
    "fmt"

    "github.com/amzn/ion-go/ion"
    "github.com/aws/aws-sdk-go/aws"
    "github.com/aws/aws-sdk-go/aws/session"
    "github.com/aws/aws-sdk-go/service/qldbsession"
    "github.com/awslabs/amazon-qldb-driver-go/v2/qldbdriver"
)

func main() {
    awsSession := session.Must(session.NewSession(aws.NewConfig().WithRegion("us-
east-1")))
    qldbSession := qldbsession.New(awsSession)

    driver, err := qldbdriver.New(
        "quick-start",
        qldbSession,
        func(options *qldbdriver.DriverOptions) {
            options.LoggerVerbosity = qldbdriver.LogInfo
        })
    if err != nil {
        panic(err)
    }
    defer driver.Shutdown(context.Background())
```

```
    _, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
(interface{}, error) {
        _, err := txn.Execute("CREATE TABLE People")
        if err != nil {
            return nil, err
        }

        // When working with QLDB, it's recommended to create an index on fields we're
filtering on.
        // This reduces the chance of OCC conflict exceptions with large datasets.
        _, err = txn.Execute("CREATE INDEX ON People (firstName)")
        if err != nil {
            return nil, err
        }

        _, err = txn.Execute("CREATE INDEX ON People (age)")
        if err != nil {
            return nil, err
        }

        return nil, nil
    })
    if err != nil {
        panic(err)
    }

    _, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
(interface{}, error) {
        return txn.Execute("INSERT INTO People {'firstName': 'Jane', 'lastName': 'Doe',
'age': 77}")
    })
    if err != nil {
        panic(err)
    }

    type Person struct {
        FirstName string `ion:"firstName"`
        LastName  string `ion:"lastName"`
        Age       int    `ion:"age"`
    }

    person := Person{"John", "Doe", 54}
```

```
    _, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
(interface{}, error) {
        return txn.Execute("INSERT INTO People ?", person)
    })
    if err != nil {
        panic(err)
    }

    p, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
(interface{}, error) {
        result, err := txn.Execute("SELECT firstName, lastName, age FROM People WHERE
age = 54")
        if err != nil {
            return nil, err
        }

        // Assume the result is not empty
        hasNext := result.Next(txn)
        if !hasNext && result.Err() != nil {
            return nil, result.Err()
        }

        ionBinary := result.GetCurrentData()

        temp := new(Person)
        err = ion.Unmarshal(ionBinary, temp)
        if err != nil {
            return nil, err
        }

        return *temp, nil
    })
    if err != nil {
        panic(err)
    }

    var returnedPerson Person
    returnedPerson = p.(Person)

    if returnedPerson != person {
        fmt.Print("Queried result does not match inserted struct")
    }

    person.Age += 10
```

```
   _, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
(interface{}, error) {
       return txn.Execute("UPDATE People SET age = ? WHERE firstName = ?", person.Age,
person.FirstName)
   })
   if err != nil {
       panic(err)
   }

   p, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
(interface{}, error) {
       result, err := txn.Execute("SELECT firstName, lastName, age FROM People WHERE
firstName = ?", person.FirstName)
       if err != nil {
           return nil, err
       }

       var people []Person
       for result.Next(txn) {
           ionBinary := result.GetCurrentData()

           temp := new(Person)
           err = ion.Unmarshal(ionBinary, temp)
           if err != nil {
               return nil, err
           }

           people = append(people, *temp)
       }
       if result.Err() != nil {
           return nil, result.Err()
       }

       return people, nil
   })
   if err != nil {
       panic(err)
   }

   var people []Person
   people = p.([]Person)

   updatedPerson := Person{"John", "Doe", 64}
```

```
    if people[0] != updatedPerson {
        fmt.Print("Queried result does not match updated struct")
    }

    _, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
        return txn.Execute("DROP TABLE People")
    })
    if err != nil {
        panic(err)
    }
}
```

Pilote Amazon QLDB pour Go — Référence du livre de recettes

Ce guide de référence présente des cas d'utilisation courants du pilote Amazon QLDB pour Go. Il fournit des exemples de code Go montrant comment utiliser le pilote pour exécuter des opérations de base de création, lecture, mise à jour et suppression (CRUD) de base. Il inclut également des exemples de code pour le traitement des données Amazon Ion. En outre, ce guide met en évidence les meilleures pratiques pour rendre les transactions idempotentes et mettre en œuvre des contraintes d'unicité.

> **Note**
>
> Le cas échéant, certains cas d'utilisation comportent des exemples de code différents pour chaque version majeure prise en charge du pilote QLDB pour Go.

Table des matières

Importation du pilote

L'exemple de code suivant importe le pilote et les autresAWS packages requis.

3.x

```
import (

    "github.com/amzn/ion-go/ion"
    "github.com/aws/aws-sdk-go/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/qldbSession"
    "github.com/awslabs/amazon-qldb-driver-go/v3/qldbdriver"
)
```

2.x

```
import (

    "github.com/amzn/ion-go/ion"
    "github.com/aws/aws-sdk-go/aws"
    "github.com/aws/aws-sdk-go/aws/session"
    "github.com/aws/aws-sdk-go/service/qldbsession"
    "github.com/awslabs/amazon-qldb-driver-go/v2/qldbdriver"
)
```

> **Note**
>
> Cet exemple importe également le package Amazon Ion (`amzn/ion-go/ion`). Vous avez besoin de ce package pour traiter les données Ion lors de l'exécution de certaines opérations de données dans cette référence. Pour en savoir plus, consultez Utilisation d'Amazon Ion.

Instanciation du pilote

L'exemple de code suivant crée une instance du pilote qui se connecte à un nom de registre spécifié dans un registre spécifiéRégion AWS.

3.x

```
cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    panic(err)
}

qldbSession := qldbsession.NewFromConfig(cfg, func(options *qldbsession.Options) {
    options.Region = "us-east-1"
})
driver, err := qldbdriver.New(
  "vehicle-registration",
  qldbSession,
  func(options *qldbdriver.DriverOptions) {
    options.LoggerVerbosity = qldbdriver.LogInfo
})
if err != nil {
  panic(err)
}

defer driver.Shutdown(context.Background())
```

2.x

```
awsSession := session.Must(session.NewSession(aws.NewConfig().WithRegion("us-
east-1")))
qldbSession := qldbsession.New(awsSession)

driver, err := qldbdriver.New(
```

```
  "vehicle-registration",
  qldbSession,
  func(options *qldbdriver.DriverOptions) {
    options.LoggerVerbosity = qldbdriver.LogInfo
  })
if err != nil {
  panic(err)
}
```

Opérations CRUD

QLDB exécute des opérations de création, lecture, mise à jour et suppression (CRUD) dans le cadre d'une transaction.

> **Warning**
>
> À titre de bonne pratique, vos transactions d'écriture sont strictement idempotentes.

Rendre les transactions idempotentes

Nous vous recommandons de rendre les transactions d'écriture idempotentes afin d'éviter tout effet secondaire inattendu en cas de nouvelles tentatives. Une transaction est idempotente si elle peut être exécutée plusieurs fois et produire des résultats identiques à chaque fois.

Prenons l'exemple d'une transaction qui insère un document dans une table nomméePerson. La transaction doit d'abord vérifier si le document existe déjà dans le tableau. Sans cette vérification, le tableau risque de contenir des documents dupliqués.

Supposons que QLDB valide avec succès la transaction côté serveur, mais que le client expire en attendant une réponse. Si la transaction n'est pas idempotente, le même document peut être inséré plusieurs fois en cas de nouvelle tentative.

Utilisation d'index pour éviter l'analyse complète des tables

Nous vous recommandons également d'exécuter des instructions avec une clause deWHERE prédicat à l'aide d'un opérateur d'égalité sur un champ indexé ou un identifiant de document, par exemple,`WHERE indexedField = 123` ou`WHERE indexedField IN (456, 789)`. Sans cette recherche indexée, QLDB doit effectuer une analyse des tables, ce qui peut entraîner des délais d'expiration des transactions ou des conflits de contrôle optimiste de la concurrence (OCC).

Pour plus d'informations sur l'OCC, consultezModèle de mise QLDB simultansimultansimultansimultansimultansimultansimultansimultansimultanéité.

Transactions créées implicitement

La fonction QLDBDriver.Execute accepte une fonction lambda qui reçoit une instance de Transaction, que vous pouvez utiliser pour exécuter des instructions. L'instance de`Transaction` encapsule une transaction créée implicitement.

Vous pouvez exécuter des instructions dans la fonction Lambda à l'aide de cette`Transaction.Execute` fonction. Le pilote valide implicitement la transaction lorsque la fonction Lambda est renvoyée.

Les sections suivantes montrent comment exécuter des opérations CRUD de base, spécifier une logique de nouvelle tentative personnalisée et implémenter des contraintes d'unicité.

Table des matières

Création de tables

```
result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  return txn.Execute("CREATE TABLE Person")
})
```

Création d'index

```
result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  return txn.Execute("CREATE INDEX ON Person(GovId)")
})
```

Lecture de documents

```
var decodedResult map[string]interface{}

// Assumes that Person table has documents as follows:
// { "GovId": "TOYENC486FH", "FirstName": "Brent" }
_, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  result, err := txn.Execute("SELECT * FROM Person WHERE GovId = 'TOYENC486FH'")
  if err != nil {
    return nil, err
  }
  for result.Next(txn) {
    ionBinary := result.GetCurrentData()
    err = ion.Unmarshal(ionBinary, &decodedResult)
    if err != nil {
      return nil, err
    }
    fmt.Println(decodedResult) // prints map[GovId: TOYENC486FH FirstName:Brent]
  }
  if result.Err() != nil {
    return nil, result.Err()
  }
  return nil, nil
})
if err != nil {
  panic(err)
}
```

Utilisation des paramètres de requête

L'exemple de code suivant utilise un paramètre de requête de type natif.

```
result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
```

```
  return txn.Execute("SELECT * FROM Person WHERE GovId = ?", "TOYENC486FH")
})
if err != nil {
  panic(err)
}
```

L'exemple de code suivant utilise plusieurs paramètres de requête.

```
result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  return txn.Execute("SELECT * FROM Person WHERE GovId = ? AND FirstName = ?",
 "TOYENC486FH", "Brent")
})
if err != nil {
  panic(err)
}
```

L'exemple de code suivant utilise une liste de paramètres de requête.

```
govIDs := []string{}{"TOYENC486FH", "ROEE1", "YH844"}

result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  return txn.Execute("SELECT * FROM Person WHERE GovId IN (?,?,?)", govIDs...)
})
if err != nil {
  panic(err)
}
```

> **Note**
>
> Lorsque vous exécutez une requête sans recherche indexée, elle appelle une analyse complète de la table. Dans cet exemple, nous recommandons de disposer d'un index sur le`GovId` terrain afin d'optimiser les performances. Si aucun index n'est activé`GovId`, les requêtes peuvent présenter une latence plus importante et peuvent également entraîner des exceptions de conflit OCC ou des délais de transaction.

Insertion de documents

L'exemple de code suivant insère des types de données natifs.

```
_, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  // Check if a document with a GovId of TOYENC486FH exists
  // This is critical to make this transaction idempotent
  result, err := txn.Execute("SELECT * FROM Person WHERE GovId = ?", "TOYENC486FH")
  if err != nil {
    return nil, err
  }
  // Check if there are any results
  if result.Next(txn) {
    // Document already exists, no need to insert
  } else {
    person := map[string]interface{}{
      "GovId": "TOYENC486FH",
      "FirstName": "Brent",
    }
    _, err = txn.Execute("INSERT INTO Person ?", person)
    if err != nil {
      return nil, err
    }
  }
  return nil, nil
})
```

Cette transaction insère un document dans le`Person` tableau. Avant de l'insérer, il vérifie d'abord si le document existe déjà dans le tableau. Ce contrôle confère à la transaction un caractère idempotent. Même si vous exécutez cette transaction plusieurs fois, elle n'aura aucun effet secondaire imprévu.

> **Note**
>
> Dans cet exemple, nous recommandons de disposer d'un index sur le`GovId` terrain afin d'optimiser les performances. Si aucun index n'est activé`GovId`, les instructions peuvent présenter une latence plus importante et peuvent également entraîner des exceptions de conflit OCC ou des délais de transaction.

Insertion de plusieurs documents dans une seule déclaration

Pour insérer plusieurs documents à l'aide d'une seuleINSERT instruction, vous pouvez transmettre un paramètre de type list à l'instruction comme suit.

```
// people is a list
txn.Execute("INSERT INTO People ?", people)
```

Vous ne placez pas l'espace réservé à la variable (?) entre crochets (`<<...>>`) lorsque vous transmettez une liste. Dans les instructions manuelles de PartiQL, les crochets doubles indiquent une collection non ordonnée appelée sac.

Mise à jour des documents

L'exemple de code suivant utilise des types de données natifs.

```
result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  return txn.Execute("UPDATE Person SET FirstName = ? WHERE GovId = ?", "John",
 "TOYENC486FH")
})
```

> **Note**
>
> Dans cet exemple, nous recommandons de disposer d'un index sur le`GovId` terrain afin d'optimiser les performances. Si aucun index n'est activé`GovId`, les instructions peuvent présenter une latence plus importante et peuvent également entraîner des exceptions de conflit OCC ou des délais de transaction.

Supprimer des documents

L'exemple de code suivant utilise des types de données natifs.

```
result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  return txn.Execute("DELETE FROM Person WHERE GovId = ?", "TOYENC486FH")
})
```

> **Note**
>
> Dans cet exemple, nous recommandons de disposer d'un index sur le`GovId` terrain afin d'optimiser les performances. Si aucun index n'est activé`GovId`, les instructions peuvent

présenter une latence plus importante et peuvent également entraîner des exceptions de conflit OCC ou des délais de transaction.

Exécution de plusieurs relevés dans le cadre d'une transaction

```
// This code snippet is intentionally trivial. In reality you wouldn't do this because
 you'd
// set your UPDATE to filter on vin and insured, and check if you updated something or
 not.
func InsureCar(driver *qldbdriver.QLDBDriver, vin string) (bool, error) {
    insured, err := driver.Execute(context.Background(), func(txn
 qldbdriver.Transaction) (interface{}, error) {

        result, err := txn.Execute(
            "SELECT insured FROM Vehicles WHERE vin = ? AND insured = FALSE", vin)
        if err != nil {
            return false, err
        }

        hasNext := result.Next(txn)
        if !hasNext && result.Err() != nil {
            return false, result.Err()
        }

        if hasNext {
            _, err = txn.Execute(
                "UPDATE Vehicles SET insured = TRUE WHERE vin = ?", vin)
            if err != nil {
                return false, err
            }
            return true, nil
        }
        return false, nil
    })
    if err != nil {
        panic(err)
    }

    return insured.(bool), err
}
```

Logique de nouvelles tentatives

La`Execute` fonction du pilote possède un mécanisme de nouvelle tentative intégré qui réessaie la transaction si une exception pouvant être réessayée se produit (par exemple, des délais d'attente ou des conflits OCC). Le nombre maximum de nouvelles tentatives maximales et la stratégie d'arrêt sont configurables.

La limite de tentatives par défaut est de`4`, et la stratégie d'annulation par défaut est [ExponentialBackoffStrategy](#)basée sur des`10` millisecondes. Vous pouvez définir la politique de nouvelle tentative par instance de pilote et également par transaction en utilisant une instance de [RetryPolicy](#).

L'exemple de code suivant spécifie une logique de nouvelle tentative avec une limite de tentatives personnalisée et une stratégie d'arrêt personnalisée pour une instance du pilote.

```
import (
  "github.com/aws/aws-sdk-go/aws"
  "github.com/aws/aws-sdk-go/aws/session"
  "github.com/aws/aws-sdk-go/service/qldbsession"
  "github.com/awslabs/amazon-qldb-driver-go/v2/qldbdriver"
)

func main() {
  awsSession := session.Must(session.NewSession(aws.NewConfig().WithRegion("us-
east-1")))
  qldbSession := qldbsession.New(awsSession)

  // Configuring retry limit to 2
  retryPolicy := qldbdriver.RetryPolicy{MaxRetryLimit: 2}

  driver, err := qldbdriver.New("test-ledger", qldbSession, func(options
 *qldbdriver.DriverOptions) {
    options.RetryPolicy = retryPolicy
  })
  if err != nil {
    panic(err)
  }

  // Configuring an exponential backoff strategy with base of 20 milliseconds
  retryPolicy = qldbdriver.RetryPolicy{
    MaxRetryLimit: 2,
    Backoff: qldbdriver.ExponentialBackoffStrategy{SleepBase: 20, SleepCap: 4000,
```

```
    }}

  driver, err = qldbdriver.New("test-ledger", qldbSession, func(options
 *qldbdriver.DriverOptions) {
    options.RetryPolicy = retryPolicy
  })
  if err != nil {
    panic(err)
  }
}
```

L'exemple de code suivant spécifie une logique de nouvelle tentative avec une limite de tentatives personnalisée et une stratégie d'arrêt personnalisée pour une fonction anonyme particulière. La`SetRetryPolicy` fonction remplace la politique de nouvelle tentative définie pour l'instance du pilote.

```
import (
  "context"
  "github.com/aws/aws-sdk-go/aws"
  "github.com/aws/aws-sdk-go/aws/session"
  "github.com/aws/aws-sdk-go/service/qldbsession"
  "github.com/awslabs/amazon-qldb-driver-go/v2/qldbdriver"
)

func main() {
  awsSession := session.Must(session.NewSession(aws.NewConfig().WithRegion("us-
east-1")))
  qldbSession := qldbsession.New(awsSession)

  // Configuring retry limit to 2
  retryPolicy1 := qldbdriver.RetryPolicy{MaxRetryLimit: 2}

  driver, err := qldbdriver.New("test-ledger", qldbSession, func(options
 *qldbdriver.DriverOptions) {
    options.RetryPolicy = retryPolicy1
  })
  if err != nil {
    panic(err)
  }

  // Configuring an exponential backoff strategy with base of 20 milliseconds
  retryPolicy2 := qldbdriver.RetryPolicy{
    MaxRetryLimit: 2,
```

```
    Backoff: qldbdriver.ExponentialBackoffStrategy{SleepBase: 20, SleepCap: 4000,
    }}

  // Overrides the retry policy set by the driver instance
  driver.SetRetryPolicy(retryPolicy2)

  driver.Execute(context.Background(), func(txn qldbdriver.Transaction) (interface{},
 error) {
    return txn.Execute("CREATE TABLE Person")
  })
}
```

Mise en œuvre des contraintes d'unicité

QLDB ne prend pas en charge les index uniques, mais vous pouvez implémenter ce comportement dans votre application.

Supposons que vous souhaitiez implémenter une contrainte d'unicité sur le`GovId` champ de la`Person` table. Pour ce faire, vous pouvez écrire une transaction qui effectue les actions suivantes :

1. Affirmez que la table ne contient aucun document existant portant une valeur spécifiée`GovId`.
2. Insérez le document si l'assertion est acceptée.

Si une transaction concurrente passe simultanément l'assertion, seule l'une des transactions sera validée avec succès. L'autre transaction échouera avec une exception de conflit OCC.

L'exemple de code suivant montre comment implémenter cette logique de contrainte d'unicité.

```
govID := "TOYENC486FH"

document := map[string]interface{}{
  "GovId":     "TOYENC486FH",
  "FirstName": "Brent",
}

result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  // Check if doc with GovId = govID exists
  result, err := txn.Execute("SELECT * FROM Person WHERE GovId = ?", govID)
  if err != nil {
    return nil, err
  }
```

```
  // Check if there are any results
  if result.Next(txn) {
    // Document already exists, no need to insert
    return nil, nil
  }
  return txn.Execute("INSERT INTO Person ?", document)
})
if err != nil {
  panic(err)
}
```

> **Note**
>
> Dans cet exemple, nous recommandons de disposer d'un index sur le`GovId` terrain afin d'optimiser les performances. Si aucun index n'est activé`GovId`, les instructions peuvent présenter une latence plus importante et peuvent également entraîner des exceptions de conflit OCC ou des délais de transaction.

Utilisation d'Amazon Ion

Les sections suivantes montrent comment utiliser le module Amazon Ion pour traiter les données Ion.

Table des matières

Importation du module Ion

```
import "github.com/amzn/ion-go/ion"
```

Création de types Ion

La bibliothèque Ion pour Go ne prend actuellement pas en charge le modèle DOM (Document Object Model). Vous ne pouvez donc pas créer de types de données Ion. Mais vous pouvez organiser et dégrouper entre les types natifs de Go et le binaire Ion lorsque vous travaillez avec QLDB.

Obtention d'Ion

```
aDict := map[string]interface{}{
  "GovId": "TOYENC486FH",
  "FirstName": "Brent",
}

ionBytes, err := ion.MarshalBinary(aDict)
if err != nil {
  panic(err)
}

fmt.Println(ionBytes) // prints [224 1 0 234 238 151 129 131 222 147 135 190 144 133 71
 111 118 73 100 137 70 105 114 115 116 78 97 109 101 222 148 138 139 84 79 89 69 78 67
 52 56 54 70 72 139 133 66 114 101 110 116]
```

Obtention d'Ion

```
aDict := map[string]interface{}{
  "GovId": "TOYENC486FH",
  "FirstName": "Brent",
}

ionBytes, err := ion.MarshalText(aDict)
if err != nil {
  panic(err)
}

fmt.Println(string(ionBytes)) // prints {FirstName:"Brent",GovId:"TOYENC486FH"}
```

Pour plus d'informations sur Ion, consultez la documentation Amazon Ion sur GitHub. Pour plus d'exemples de code illustrant l'utilisation d'Ion dans QLDB, consultezUtilisation des types de données Amazon Ion dans Amazon QLDB.

pilote Amazon QLDB pour Node.js

Pour utiliser les données de votre registre, vous pouvez vous connecter à Amazon QLDB depuis votre application Node.js à l'aide d'un pilote fourni. AWS Les rubriques suivantes décrivent comment démarrer avec le pilote QLDB pour Node.js.

Rubriques

Ressources pour les conducteurs

Pour plus d'informations sur les fonctionnalités prises en charge par le pilote Node.js, consultez les ressources suivantes :

- Référence d'API : 3.x, 2.x, 1.x
- Code source du pilote (GitHub)
- Exemple de code source d'application (GitHub)
- Exemples de code Amazon Ion
- Créez une opération CRUD et un flux de données simples sur AWS Lambda QLDB en utilisant (Blog) AWS

Prérequis

Avant de commencer à utiliser le pilote QLDB pour Node.js, vous devez effectuer les opérations suivantes :

1. Suivez les instructions AWS de configuration indiquées dansAccès à Amazon QLDB. Cela inclut les éléments suivants :
    1. Inscrivez-vous pourAWS.
    2. Créez un utilisateur doté des autorisations QLDB appropriées.
    3. Accordez un accès programmatique pour le développement.
2. Installez Node.js version 14.x ou ultérieure à partir du site de téléchargement de Node.js. (Les versions précédentes du pilote prennent en charge la version 10.x ou ultérieure de Node.js.)
3. Configurez votre environnement de développement pour le AWSSDK JavaScript dans Node.js :

1. Configurez vos AWS informations d'identification. Nous vous recommandons de créer un fichier d'informations d'identification partagé.

   Pour obtenir des instructions, consultez la section Chargement des informations d'identification dans le fichier Node.js à partir du fichier d'informations d'identification partagé du manuel du AWS SDK for JavaScript développeur.

2. Définissez votre valeur par défautRégion AWS. Pour savoir comment procéder, reportez-vous à la section Configuration du Région AWS.

   Pour obtenir la liste complète des régions disponibles, consultez la section Points de terminaison et quotas Amazon QLDB dans le. Références générales AWS

Ensuite, vous pouvez télécharger l'exemple d'application complet du didacticiel, ou vous pouvez installer uniquement le pilote dans un projet Node.js et exécuter des exemples de code abrégé.

- Pour installer le pilote QLDB et AWS le SDK JavaScript pour in Node.js dans un projet existant, passez à. Installation
- Pour configurer un projet et exécuter des exemples de codes abrégés illustrant les transactions de données de base sur un registre, consultez leDidacticiel de démarrage rapide.
- Pour obtenir des exemples plus détaillés des opérations relatives aux données et aux API de gestion dans l'exemple d'application complet du didacticiel, consultez leTutoriel Node.js.

Installation

QLDB prend en charge les versions de pilotes suivantes et leurs dépendances avec Node.js.

Versions du pilote	Node.js version (Version de Node.js)	Statut	Date de sortie la plus récente
1. x	10.x ou version ultérieure	Communiqué de production	5 juin 2020
2. x	10.x ou version ultérieure	Communiqué de production	6 mai 2021

Versions du pilote	Node.js version (Version de Node.js)	Statut	Date de sortie la plus récente
3. x	14.x ou version ultérieure	Communiqué de production	10 novembre 2023

Pour installer le pilote QLDB à l'aide de npm (le gestionnaire de packages Node.js), entrez la commande suivante depuis le répertoire racine de votre projet.

3.x

```
npm install amazon-qldb-driver-nodejs
```

2.x

```
npm install amazon-qldb-driver-nodejs@2.2.0
```

1.x

```
npm install amazon-qldb-driver-nodejs@1.0.0
```

Le pilote a des dépendances entre pairs sur les packages suivants. Vous devez également installer ces packages en tant que dépendances dans votre projet.

3.x

Client QLDB agrégé modulaire (API de gestion)

```
npm install @aws-sdk/client-qldb
```

Client de session QLDB agrégé modulaire (API de données)

```
npm install @aws-sdk/client-qldb-session
```

Format de données Amazon Ion

```
npm install ion-js
```

JavaScript Mise en œuvre pure de `BigInt`

```
npm install jsbi
```

2.x

AWS SDK for JavaScript

```
npm install aws-sdk
```

Format de données Amazon Ion

```
npm install ion-js@4.0.0
```

JavaScript Mise en œuvre pure de `BigInt`

```
npm install jsbi@3.1.1
```

1.x

AWS SDK for JavaScript

```
npm install aws-sdk
```

Format de données Amazon Ion

```
npm install ion-js@4.0.0
```

JavaScript Mise en œuvre pure de `BigInt`

```
npm install jsbi@3.1.1
```

Utiliser le pilote pour se connecter à un registre

Vous pouvez ensuite importer le pilote et l'utiliser pour vous connecter à un registre. L'exemple de TypeScript code suivant montre comment créer une instance de pilote pour un nom de registre spécifié etRégion AWS.

3.x

```
import { Agent } from 'https';
import { QLDBSessionClientConfig } from "@aws-sdk/client-qldb-session";
import { QldbDriver, RetryConfig  } from 'amazon-qldb-driver-nodejs';
import { NodeHttpHandlerOptions } from "@aws-sdk/node-http-handler";

const maxConcurrentTransactions: number = 10;
const retryLimit: number = 4;

//Reuse connections with keepAlive
const lowLevelClientHttpOptions: NodeHttpHandlerOptions = {
    httpAgent: new Agent({
      maxSockets: maxConcurrentTransactions
    })
};

const serviceConfigurationOptions: QLDBSessionClientConfig = {
    region: "us-east-1"
};

//Use driver's default backoff function for this example (no second parameter
 provided to RetryConfig)
const retryConfig: RetryConfig = new RetryConfig(retryLimit);
const qldbDriver: QldbDriver = new QldbDriver("testLedger",
 serviceConfigurationOptions, lowLevelClientHttpOptions, maxConcurrentTransactions,
 retryConfig);

qldbDriver.getTableNames().then(function(tableNames: string[]) {
    console.log(tableNames);
});
```

2.x

```
import { Agent } from 'https';
import { QldbDriver, RetryConfig  } from 'amazon-qldb-driver-nodejs';

const maxConcurrentTransactions: number = 10;
const retryLimit: number = 4;

//Reuse connections with keepAlive
const agentForQldb: Agent = new Agent({
    keepAlive: true,
```

```
    maxSockets: maxConcurrentTransactions
});

const serviceConfigurationOptions = {
    region: "us-east-1",
    httpOptions: {
        agent: agentForQldb
    }
};

//Use driver's default backoff function for this example (no second parameter
 provided to RetryConfig)
const retryConfig: RetryConfig = new RetryConfig(retryLimit);
const qldbDriver: QldbDriver = new QldbDriver("testLedger",
 serviceConfigurationOptions, maxConcurrentTransactions, retryConfig);

qldbDriver.getTableNames().then(function(tableNames: string[]) {
    console.log(tableNames);
});
```

1.x

```
import { Agent } from 'https';
import { QldbDriver } from 'amazon-qldb-driver-nodejs';

const poolLimit: number = 10;
const retryLimit: number = 4;

//Reuse connections with keepAlive
const agentForQldb: Agent = new Agent({
    keepAlive: true,
    maxSockets: poolLimit
});

const serviceConfigurationOptions = {
    region: "us-east-1",
    httpOptions: {
        agent: agentForQldb
    }
};

const qldbDriver: QldbDriver = new QldbDriver("testLedger",
 serviceConfigurationOptions, retryLimit, poolLimit);
```

```
qldbDriver.getTableNames().then(function(tableNames: string[]) {
    console.log(tableNames);
});
```

Pour des exemples de code abrégé expliquant comment exécuter des transactions de données de base sur un registre, consultez leRéférence de livre de cuisine.

Recommandations de configuration

Réutilisation des connexions avec keep-alive

pilote QLDB Node.js v3

L'agent HTTP/HTTPS Node.js par défaut crée une nouvelle connexion TCP pour chaque nouvelle demande. Pour éviter les coûts liés à l'établissement d'une nouvelle connexion, la AWS SDK for JavaScript version 3 réutilise les connexions TCP par défaut. Pour plus d'informations et pour savoir comment désactiver la réutilisation des connexions, consultez la section Réutilisation des connexions avec keep-alive dans le fichier Node.js du manuel du développeur. AWS SDK for JavaScript

Nous vous recommandons d'utiliser le paramètre par défaut pour réutiliser les connexions dans le pilote QLDB pour Node.js. Lors de l'initialisation du pilote, définissez l'option HTTP du client de bas niveau `maxSockets` sur la même valeur que celle que vous avez définie. `maxConcurrentTransactions`

Par exemple, consultez le TypeScript code JavaScript ou le code suivant.

JavaScript

```
const qldb = require('amazon-qldb-driver-nodejs');
const https = require('https');

//Replace this value as appropriate for your application
const maxConcurrentTransactions = 50;

const agentForQldb = new https.Agent({
    //Set this to the same value as `maxConcurrentTransactions`(previously called
 `poolLimit`)
    //Do not rely on the default value of `Infinity`
    "maxSockets": maxConcurrentTransactions
});
```

```
const lowLevelClientHttpOptions = {
    httpAgent: agentForQldb
}

let driver = new qldb.QldbDriver("testLedger", undefined, lowLevelClientHttpOptions,
 maxConcurrentTransactions);
```

TypeScript

```
import { Agent } from 'https';
import { NodeHttpHandlerOptions } from "@aws-sdk/node-http-handler";
import { QldbDriver } from 'amazon-qldb-driver-nodejs';

//Replace this value as appropriate for your application
const maxConcurrentTransactions: number = 50;

const agentForQldb: Agent = new Agent({
    //Set this to the same value as `maxConcurrentTransactions`(previously called
 `poolLimit`)
    //Do not rely on the default value of `Infinity`
    maxSockets: maxConcurrentTransactions
});

const lowLevelClientHttpOptions: NodeHttpHandlerOptions = {
    httpAgent: agentForQldb
};

let driver = new QldbDriver("testLedger", undefined, lowLevelClientHttpOptions,
 maxConcurrentTransactions);
```

pilote QLDB Node.js v2

L’agent HTTP/HTTPS Node.js par défaut crée une nouvelle connexion TCP pour chaque nouvelle demande. Pour éviter les coûts liés à l'établissement d'une nouvelle connexion, nous vous recommandons de réutiliser une connexion existante.

Pour réutiliser les connexions dans le pilote QLDB pour Node.js, utilisez l'une des options suivantes :

- Lors de l'initialisation du pilote, définissez les options HTTP du client de bas niveau suivantes :
  - `keepAlive` – `true`

- `maxSockets` — La même valeur que celle que vous avez définie pour `maxConcurrentTransactions`

Par exemple, consultez le TypeScript code JavaScript ou le code suivant.

JavaScript

```
const qldb = require('amazon-qldb-driver-nodejs');
const https = require('https');

//Replace this value as appropriate for your application
const maxConcurrentTransactions = 50;

const agentForQldb = new https.Agent({
    "keepAlive": true,
    //Set this to the same value as `maxConcurrentTransactions`(previously called
 `poolLimit`)
    //Do not rely on the default value of `Infinity`
    "maxSockets": maxConcurrentTransactions
});

const serviceConfiguration = { "httpOptions": {
    "agent": agentForQldb
}};

let driver = new qldb.QldbDriver("testLedger", serviceConfiguration,
 maxConcurrentTransactions);
```

TypeScript

```
import { Agent } from 'https';
import { ClientConfiguration } from 'aws-sdk/clients/acm';
import { QldbDriver } from 'amazon-qldb-driver-nodejs';

//Replace this value as appropriate for your application
const maxConcurrentTransactions: number = 50;

const agentForQldb: Agent = new Agent({
    keepAlive: true,
    //Set this to the same value as `maxConcurrentTransactions`(previously called
 `poolLimit`)
    //Do not rely on the default value of `Infinity`
    maxSockets: maxConcurrentTransactions
```

```
});

const serviceConfiguration: ClientConfiguration = { httpOptions: {
    agent: agentForQldb
}};

let driver = new QldbDriver("testLedger", serviceConfiguration,
 maxConcurrentTransactions);
```

- Vous pouvez également définir la variable d'`AWS_NODEJS_CONNECTION_REUSE_ENABLED`Environnement sur`1`. Pour plus d'informations, consultez la section Réutilisation des connexions avec Keep-Alive dans Node.js dans le manuel du AWS SDK for JavaScript développeur.

> **Note**
>
> Si vous définissez cette variable d'environnement, elle affecte tous ceux Services AWS qui utilisent leAWS SDK for JavaScript.

Pilote Amazon QLDB pour Node.js — Tutoriel de démarrage rapide

Dans ce tutoriel, vous apprendrez comment configurer une application simple à l'aide du pilote Amazon QLDB pour Node.js. Ce guide comprend les étapes d'installation du pilote, JavaScript ainsi que TypeScript des exemples de base de création, lecture, mise à jour et suppression (CRUD). Pour des exemples plus détaillés illustrant ces opérations dans un exemple d'application complet, consultez leTutoriel Node.js.

> **Note**
>
> Le cas échéant, certaines étapes comportent des exemples de code différents pour chaque version majeure prise en charge du pilote QLDB pour Node.js.

Rubriques

Prérequis

Avant de commencer, veillez à exécuter les actions suivantes :

1. Complétez lePrérequis pour le pilote Node.js, si vous ne l'avez pas déjà fait. Cela inclut l'inscriptionAWS, l'octroi d'un accès programmatique pour le développement et l'installation de Node.js.
2. Créez un registre nomméquick-start.

   Pour savoir comment créer un registre, consultezOpérations de base pour les registres Amazon QLDB ouÉtape 1 : Créer un nouveau registre dans Prise en main de la console.

Si vous utilisez TypeScript, vous devez également suivre les étapes de configuration suivantes.

En utilisant TypeScript

Pour installer TypeScript

1. Installez le TypeScript package. Le pilote QLDB s'exécute sur TypeScript 3.8.x.

   ```
   $ npm install --global typescript@3.8.0
   ```

2. Une fois le package installé, exécutez la commande suivante pour vous assurer que le TypeScript compilateur est installé.

   ```
   $ tsc --version
   ```

Pour exécuter le code dans les étapes suivantes, notez que vous devez d'abord transpiler votre TypeScript fichier en JavaScript code exécutable, comme suit.

```
$ tsc app.ts; node app.js
```

Étape 1 : Configurer votre projet

Tout d'abord, configurez votre projet Node.js.

1. Créez un dossier pour votre application.

```
$ mkdir myproject
$ cd myproject
```

2. Pour initialiser votre projet, entrez lanpm commande suivante et répondez aux questions qui vous sont posées lors de la configuration. Vous pouvez utiliser les valeurs par défaut pour la plupart des questions.

```
$ npm init
```

3. Installez le pilote Amazon QLDB pour Node.js.
   - Utilisation de la version 3.x

```
$ npm install amazon-qldb-driver-nodejs --save
```

   - Utilisation de la version 2.x

```
$ npm install amazon-qldb-driver-nodejs@2.2.0 --save
```

   - Utilisation de la version 1.x

```
$ npm install amazon-qldb-driver-nodejs@1.0.0 --save
```

4. Installez les dépendances homologues du pilote.
   - Utilisation de la version 3.x

```
$ npm install @aws-sdk/client-qldb-session --save
$ npm install ion-js --save
$ npm install jsbi --save
```

   - Utilisation de la version 2.x ou 1.x

```
$ npm install aws-sdk --save
$ npm install ion-js@4.0.0 --save
$ npm install jsbi@3.1.1 --save
```

5. Créez un nouveau fichier nomméapp.js pour JavaScript, ouapp.ts pour TypeScript.

   Ajoutez ensuite progressivement les exemples de code dans les étapes suivantes pour essayer certaines opérations CRUD de base. Vous pouvez également ignorer le step-by-step didacticiel et exécuter l'application complète.

Étape 2 : Initialiser le pilote

Initialisez une instance du pilote qui se connecte au registre nomméquick-start. Ajoutez le code suivant à votreapp.ts fichierapp.js or.

Utilisation de la version 3.x

JavaScript

```
var qldb = require('amazon-qldb-driver-nodejs');
var https = require('https');

function main() {
    const maxConcurrentTransactions = 10;
    const retryLimit = 4;

    const agentForQldb = new https.Agent({
        maxSockets: maxConcurrentTransactions
    });

    const lowLevelClientHttpOptions = {
       httpAgent: agentForQldb
    }

    const serviceConfigurationOptions = {
        region: "us-east-1"
    };

    // Use driver's default backoff function for this example (no second parameter
 provided to RetryConfig)
    var retryConfig = new qldb.RetryConfig(retryLimit);
    var driver = new qldb.QldbDriver("quick-start", serviceConfigurationOptions,
 lowlevelClientHttpOptions, maxConcurrentTransactions, retryConfig);
}

main();
```

TypeScript

```
import { Agent } from "https";
import { NodeHttpHandlerOptions } from "@aws-sdk/node-http-handler";
import { QLDBSessionClientConfig } from "@aws-sdk/client-qldb-session";
import { QldbDriver, RetryConfig } from "amazon-qldb-driver-nodejs";

function main(): void {
    const maxConcurrentTransactions: number = 10;
    const agentForQldb: Agent = new Agent({
        maxSockets: maxConcurrentTransactions
    });

    const lowLevelClientHttpOptions: NodeHttpHandlerOptions = {
      httpAgent: agentForQldb
    };

    const serviceConfigurationOptions: QLDBSessionClientConfig = {
        region: "us-east-1"
    };

    const retryLimit: number = 4;
    // Use driver's default backoff function for this example (no second parameter
 provided to RetryConfig)
    const retryConfig: RetryConfig = new RetryConfig(retryLimit);
    const driver: QldbDriver = new QldbDriver("quick-start",
 serviceConfigurationOptions, lowLevelClientHttpOptions, maxConcurrentTransactions,
 retryConfig);
}

if (require.main === module) {
    main();
}
```

> **Note**
>
> - Dans cet exemple de code, remplacez *us-east-1* par l'Région AWSendroit où vous avez créé votre registre.
> - Pour des raisons de simplicité, les autres exemples de code de ce guide utilisent un pilote avec des paramètres par défaut, comme indiqué dans l'exemple suivant pour la

version 1.x. Vous pouvez également utiliser votre propre instance de pilote avec une instance`RetryConfig` personnalisée.

Utilisation de la version 2.x

JavaScript

```
var qldb = require('amazon-qldb-driver-nodejs');
var https = require('https');

function main() {
    var maxConcurrentTransactions = 10;
    var retryLimit = 4;

    var agentForQldb = new https.Agent({
        keepAlive: true,
        maxSockets: maxConcurrentTransactions
    });

    var serviceConfigurationOptions = {
        region: "us-east-1",
        httpOptions: {
            agent: agentForQldb
        }
    };

    // Use driver's default backoff function for this example (no second parameter
 provided to RetryConfig)
    var retryConfig = new qldb.RetryConfig(retryLimit);
    var driver = new qldb.QldbDriver("quick-start", serviceConfigurationOptions,
 maxConcurrentTransactions, retryConfig);
}

main();
```

TypeScript

```
import { QldbDriver, RetryConfig } from "amazon-qldb-driver-nodejs";
import { ClientConfiguration } from "aws-sdk/clients/acm";
import { Agent } from "https";
```

```
function main(): void {
    const maxConcurrentTransactions: number = 10;
    const agentForQldb: Agent = new Agent({
        keepAlive: true,
        maxSockets: maxConcurrentTransactions
    });
    const serviceConfigurationOptions: ClientConfiguration = {
        region: "us-east-1",
        httpOptions: {
            agent: agentForQldb
        }
    };
    const retryLimit: number = 4;
    // Use driver's default backoff function for this example (no second parameter
 provided to RetryConfig)
    const retryConfig: RetryConfig = new RetryConfig(retryLimit);
    const driver: QldbDriver = new QldbDriver("quick-start",
 serviceConfigurationOptions, maxConcurrentTransactions, retryConfig);
}

if (require.main === module) {
    main();
}
```

Note

- Dans cet exemple de code, remplacez *us-east-1* par l'Région AWSendroit où vous avez créé votre registre.
- La version 2.x introduit le nouveau paramètre optionnel`RetryConfig` pour l'initialisation`QldbDriver`.
- Pour des raisons de simplicité, les autres exemples de code de ce guide utilisent un pilote avec des paramètres par défaut, comme indiqué dans l'exemple suivant pour la version 1.x. Vous pouvez également utiliser votre propre instance de pilote avec une instance`RetryConfig` personnalisée.
- Cet exemple de code initialise un pilote qui réutilise les connexions existantes en définissant des options keep-alive. Pour de plus amples informations,Recommandations de configuration veuillez consulter le pilote Node.js.

Utilisation de la version 1.x

JavaScript

```
const qldb = require('amazon-qldb-driver-nodejs');

function main() {
    // Use default settings
    const driver = new qldb.QldbDriver("quick-start");
}

main();
```

TypeScript

```
import { QldbDriver } from "amazon-qldb-driver-nodejs";

function main(): void {
    // Use default settings
    const driver: QldbDriver = new QldbDriver("quick-start");
}

if (require.main === module) {
    main();
}
```

> **Note**
>
> Vous pouvez définir la variable d'`AWS_REGION`environnement pour spécifier la région. Pour plus d'informations, consultez la section Configuration duRégion AWS dans le Guide duAWS SDK for JavaScript développeur.

Étape 3 : création d'une table et table

Les exemples de code suivants montrent comment exécuter`CREATE INDEX` des instructions`CREATE TABLE` and.

1. Ajoutez la fonction suivante qui crée une table nommée`People`.

JavaScript

```
async function createTable(txn) {
    await txn.execute("CREATE TABLE People");
}
```

TypeScript

```
async function createTable(txn: TransactionExecutor): Promise<void> {
    await txn.execute("CREATE TABLE People");
}
```

2. Ajoutez la fonction suivante qui crée un index pour le`firstName` champ de la`People` table. Les index sont nécessaires pour optimiser les performances des requêtes et contribuer à limiter les exceptions de conflit liées au contrôle optimiste de la concurrence (OCC).

   JavaScript

   ```
   async function createIndex(txn) {
       await txn.execute("CREATE INDEX ON People (firstName)");
   }
   ```

   TypeScript

   ```
   async function createIndex(txn: TransactionExecutor): Promise<void> {
       await txn.execute("CREATE INDEX ON People (firstName)");
   }
   ```

3. Dans la`main` fonction, vous appelez d'abord`createTable`, puis vous appelez`createIndex`.

   JavaScript

   ```
   const qldb = require('amazon-qldb-driver-nodejs');

   async function main() {
       // Use default settings
       const driver = new qldb.QldbDriver("quick-start");

       await driver.executeLambda(async (txn) => {
           console.log("Create table People");
           await createTable(txn);
   ```

```
        console.log("Create index on firstName");
        await createIndex(txn);
    });

    driver.close();
}

main();
```

TypeScript

```
import { QldbDriver, TransactionExecutor } from "amazon-qldb-driver-nodejs";

async function main(): Promise<void> {
    // Use default settings
    const driver: QldbDriver = new QldbDriver("quick-start");

    await driver.executeLambda(async (txn: TransactionExecutor) => {
        console.log("Create table People");
        await createTable(txn);
        console.log("Create index on firstName");
        await createIndex(txn);
    });

    driver.close();
}

if (require.main === module) {
    main();
}
```

4. Exécutez le code pour créer la table et l'index.

JavaScript

```
$ node app.js
```

TypeScript

```
$ tsc app.ts; node app.js
```

Étape 4 : Insérer un document

L'exemple de code suivant montre comment exécuter une`INSERT` instruction. QLDB prend en charge le langage de requête PartiQL (compatible SQL) et le format de données Amazon Ion (surensemble de JSON).

1. Ajoutez la fonction suivante qui insère un document dans le`People` tableau.

   JavaScript

   ```
   async function insertDocument(txn) {
       const person = {
           firstName: "John",
           lastName: "Doe",
           age: 42
       };
       await txn.execute("INSERT INTO People ?", person);
   }
   ```

   TypeScript

   ```
   async function insertDocument(txn: TransactionExecutor): Promise<void> {
       const person: Record<string, any> = {
           firstName: "John",
           lastName: "Doe",
           age: 42
       };
       await txn.execute("INSERT INTO People ?", person);
   }
   ```

   Cet exemple utilise un point d'interrogation (?) comme espace réservé variable pour transmettre les informations du document à l'instruction. La`execute` méthode prend en charge les valeurs des types Amazon Ion et des types natifs Node.js.

   > **Tip**
   >
   > Pour insérer plusieurs documents à l'aide d'une seuleINSERT instruction, vous pouvez transmettre un paramètre de type list à l'instruction comme suit.
   >
   > ```
   > // people is a list
   > ```

```
txn.execute("INSERT INTO People ?", people);
```

Vous ne placez pas l'espace réservé à la variable (?) entre crochets (`<<...>>`) lorsque vous transmettez une liste. Dans les instructions manuelles de PartiQL, les crochets doubles indiquent une collection non ordonnée appelée sac.

2. Dans la`main` fonction, supprimez les`createIndex` appels`createTable` et, puis ajoutez un appel à`insertDocument`.

JavaScript

```
const qldb = require('amazon-qldb-driver-nodejs');

async function main() {
    // Use default settings
    const driver = new qldb.QldbDriver("quick-start");

    await driver.executeLambda(async (txn) => {
        console.log("Insert document");
        await insertDocument(txn);
    });

    driver.close();
}

main();
```

TypeScript

```
import { QldbDriver, TransactionExecutor } from "amazon-qldb-driver-nodejs";

async function main(): Promise<void> {
    // Use default settings
    const driver: QldbDriver = new QldbDriver("quick-start");

    await driver.executeLambda(async (txn: TransactionExecutor) => {
        console.log("Insert document");
        await insertDocument(txn);
    });

    driver.close();
}
```

```
if (require.main === module) {
    main();
}
```

Étape 5 : interroger le document

L'exemple de code suivant montre comment exécuter une`SELECT` instruction.

1. Ajoutez la fonction suivante qui interroge un document à partir du`People` tableau.

   JavaScript

   ```
   async function fetchDocuments(txn) {
       return await txn.execute("SELECT firstName, age, lastName FROM People WHERE
    firstName = ?", "John");
   }
   ```

   TypeScript

   ```
   async function fetchDocuments(txn: TransactionExecutor): Promise<dom.Value[]> {
       return (await txn.execute("SELECT firstName, age, lastName FROM People WHERE
    firstName = ?", "John")).getResultList();
   }
   ```

2. Dans la`main` fonction, ajoutez l'appel suivant à`fetchDocuments` après l'appel à`insertDocument`.

   JavaScript

   ```
   const qldb = require('amazon-qldb-driver-nodejs');

   async function main() {
       // Use default settings
       const driver = new qldb.QldbDriver("quick-start");

       var resultList = await driver.executeLambda(async (txn) => {
           console.log("Insert document");
           await insertDocument(txn);
           console.log("Fetch document");
           var result = await fetchDocuments(txn);
   ```

```
        return result.getResultList();
    });

    // Pretty print the result list
    console.log("The result List is ", JSON.stringify(resultList, null, 2));
    driver.close();
}

main();
```

TypeScript

```
import { QldbDriver, TransactionExecutor } from "amazon-qldb-driver-nodejs";
import { dom } from "ion-js";

async function main(): Promise<void> {
    // Use default settings
    const driver: QldbDriver = new QldbDriver("quick-start");

    const resultList: dom.Value[] = await driver.executeLambda(async (txn:
 TransactionExecutor) => {
        console.log("Insert document");
        await insertDocument(txn);
        console.log("Fetch document");
        return await fetchDocuments(txn);
    });

    // Pretty print the result list
    console.log("The result List is ", JSON.stringify(resultList, null, 2));
    driver.close();
}

if (require.main === module) {
    main();
}
```

Étape 6 : Mettre à jour le document

L'exemple de code suivant montre comment exécuter uneUPDATE instruction.

1. Ajoutez la fonction suivante qui met à jour un document dans le`People` tableau en le remplaçant`lastName` par`"Stiles"`.

JavaScript

```
async function updateDocuments(txn) {
    await txn.execute("UPDATE People SET lastName = ? WHERE firstName = ?",
 "Stiles", "John");
}
```

TypeScript

```
async function updateDocuments(txn: TransactionExecutor): Promise<void> {
    await txn.execute("UPDATE People SET lastName = ? WHERE firstName = ?",
 "Stiles", "John");
}
```

2. Dans la`main` fonction, ajoutez l'appel suivant à`updateDocuments` après l'appel à`fetchDocuments`. Ensuite, appelez à`fetchDocuments` nouveau pour voir les résultats mis à jour.

JavaScript

```
const qldb = require('amazon-qldb-driver-nodejs');

async function main() {
    // Use default settings
    const driver = new qldb.QldbDriver("quick-start");

    var resultList = await driver.executeLambda(async (txn) => {
        console.log("Insert document");
        await insertDocument(txn);
        console.log("Fetch document");
        await fetchDocuments(txn);
        console.log("Update document");
        await updateDocuments(txn);
        console.log("Fetch document after update");
        var result = await fetchDocuments(txn);
        return result.getResultList();
    });

    // Pretty print the result list
    console.log("The result List is ", JSON.stringify(resultList, null, 2));
    driver.close();
```

```
}

main();
```

TypeScript

```
import { QldbDriver, TransactionExecutor } from "amazon-qldb-driver-nodejs";
import { dom } from "ion-js";

async function main(): Promise<void> {
    // Use default settings
    const driver: QldbDriver = new QldbDriver("quick-start");

    const resultList: dom.Value[] = await driver.executeLambda(async (txn:
 TransactionExecutor) => {
        console.log("Insert document");
        await insertDocument(txn);
        console.log("Fetch document");
        await fetchDocuments(txn);
        console.log("Update document");
        await updateDocuments(txn);
        console.log("Fetch document after update");
        return await fetchDocuments(txn);
    });

    // Pretty print the result list
    console.log("The result List is ", JSON.stringify(resultList, null, 2));
    driver.close();
}

if (require.main === module) {
    main();
}
```

3. Exécutez le code pour insérer, interroger et mettre à jour un document.

JavaScript

```
$ node app.js
```

TypeScript

```
$ tsc app.ts; node app.js
```

Exécution de l'application complète

Les exemples de code suivants sont les versions complètes de`app.js` et`app.ts`. Au lieu de suivre les étapes précédentes individuellement, vous pouvez également exécuter ce code du début à la fin. Cette application montre certaines opérations CRUD de base sur le registre nommé`quick-start`.

> **Note**
>
> Avant d'exécuter ce code, assurez-vous qu'aucune table active n'est déjà nommée`People` dans le`quick-start` registre.

JavaScript

```
const qldb = require('amazon-qldb-driver-nodejs');

async function createTable(txn) {
    await txn.execute("CREATE TABLE People");
}

async function createIndex(txn) {
    await txn.execute("CREATE INDEX ON People (firstName)");
}

async function insertDocument(txn) {
    const person = {
        firstName: "John",
        lastName: "Doe",
        age: 42
    };
    await txn.execute("INSERT INTO People ?", person);
}

async function fetchDocuments(txn) {
    return await txn.execute("SELECT firstName, age, lastName FROM People WHERE
 firstName = ?", "John");
```

```
}

async function updateDocuments(txn) {
    await txn.execute("UPDATE People SET lastName = ? WHERE firstName = ?",
 "Stiles", "John");
}

async function main() {
    // Use default settings
    const driver = new qldb.QldbDriver("quick-start");

    var resultList = await driver.executeLambda(async (txn) => {
        console.log("Create table People");
        await createTable(txn);
        console.log("Create index on firstName");
        await createIndex(txn);
        console.log("Insert document");
        await insertDocument(txn);
        console.log("Fetch document");
        await fetchDocuments(txn);
        console.log("Update document");
        await updateDocuments(txn);
        console.log("Fetch document after update");
        var result = await fetchDocuments(txn);
        return result.getResultList();
    });

    // Pretty print the result list
    console.log("The result List is ", JSON.stringify(resultList, null, 2));
    driver.close();
}

main();
```

TypeScript

```
import { QldbDriver, TransactionExecutor } from "amazon-qldb-driver-nodejs";
import { dom } from "ion-js";

async function createTable(txn: TransactionExecutor): Promise<void> {
    await txn.execute("CREATE TABLE People");
}
```

```
async function createIndex(txn: TransactionExecutor): Promise<void> {
    await txn.execute("CREATE INDEX ON People (firstName)");
}

async function insertDocument(txn: TransactionExecutor): Promise<void> {
    const person: Record<string, any> = {
        firstName: "John",
        lastName: "Doe",
        age: 42
    };
    await txn.execute("INSERT INTO People ?", person);
}

async function fetchDocuments(txn: TransactionExecutor): Promise<dom.Value[]> {
    return (await txn.execute("SELECT firstName, age, lastName FROM People WHERE
 firstName = ?", "John")).getResultList();
}

async function updateDocuments(txn: TransactionExecutor): Promise<void> {
    await txn.execute("UPDATE People SET lastName = ? WHERE firstName = ?",
 "Stiles", "John");
};

async function main(): Promise<void> {
    // Use default settings
    const driver: QldbDriver = new QldbDriver("quick-start");

    const resultList: dom.Value[] = await driver.executeLambda(async (txn:
 TransactionExecutor) => {
        console.log("Create table People");
        await createTable(txn);
        console.log("Create index on firstName");
        await createIndex(txn);
        console.log("Insert document");
        await insertDocument(txn);
        console.log("Fetch document");
        await fetchDocuments(txn);
        console.log("Update document");
        await updateDocuments(txn);
        console.log("Fetch document after update");
        return await fetchDocuments(txn);
    });

    // Pretty print the result list
```

```
    console.log("The result List is ", JSON.stringify(resultList, null, 2));
    driver.close();
}

if (require.main === module) {
    main();
}
```

Pour exécuter l'application complète, saisissez la commande suivante.

JavaScript

```
$ node app.js
```

TypeScript

```
$ tsc app.ts; node app.js
```

Pilote Amazon QLDB pour Node.js — Référence du livre de recettes

Ce guide de référence présente des cas d'utilisation courants du pilote Amazon QLDB pour Node.js. Il fournit JavaScript des exemples de TypeScript code montrant comment utiliser le pilote pour effectuer des opérations de création, lecture, mise à jour et suppression (CRUD) de base. Il inclut également des exemples de code pour le traitement des données Amazon Ion. En outre, ce guide met en évidence les meilleures pratiques pour rendre les transactions idempotentes et mettre en œuvre des contraintes d'unicité.

Table des matières

Importation du pilote

L'exemple de code suivant importe le pilote.

JavaScript

```
var qldb = require('amazon-qldb-driver-nodejs');
var ionjs = require('ion-js');
```

TypeScript

```
import { QldbDriver, TransactionExecutor } from "amazon-qldb-driver-nodejs";
import { dom, dumpBinary, load } from "ion-js";
```

> **Note**
>
> Cet exemple importe également le package Amazon Ion (`ion-js`). Vous avez besoin de ce package pour traiter les données Ion lors de l'exécution de certaines opérations de données dans cette référence. Pour en savoir plus, consultez Utilisation d'Amazon Ion.

Instanciation du pilote

L'exemple de code suivant crée une instance du pilote qui se connecte à un nom de registre spécifié à l'aide des paramètres par défaut.

JavaScript

```
const qldbDriver = new qldb.QldbDriver("vehicle-registration");
```

TypeScript

```
const qldbDriver: QldbDriver = new QldbDriver("vehicle-registration");
```

Opérations CRUD

QLDB exécute des opérations de création, lecture, mise à jour et suppression (CRUD) dans le cadre d'une transaction.

> **Warning**
>
> À titre de bonne pratique, faites en sorte que vos transactions d'écriture soient strictement idempotentes.

Rendre les transactions idempotentes

Nous vous recommandons de rendre les transactions d'écriture idempotentes afin d'éviter tout effet secondaire inattendu en cas de nouvelles tentatives. Une transaction est idempotente si elle peut être exécutée plusieurs fois et produire des résultats identiques à chaque fois.

Prenons l'exemple d'une transaction qui insère un document dans une table nomméePerson. La transaction doit d'abord vérifier si le document existe déjà dans le tableau. Sans cette vérification, le tableau risque de contenir des documents dupliqués.

Supposons que QLDB valide avec succès la transaction côté serveur, mais que le client expire en attendant une réponse. Si la transaction n'est pas idempotente, le même document peut être inséré plusieurs fois en cas de nouvelle tentative.

Utilisation d'index pour éviter l'analyse complète des tables

Nous vous recommandons également d'exécuter des instructions avec une clause de`WHERE` prédicat à l'aide d'un opérateur d'égalité sur un champ indexé ou un identifiant de document, par exemple,`WHERE indexedField = 123` ou`WHERE indexedField IN (456, 789)`. Sans cette recherche indexée, QLDB doit effectuer une analyse des tables, ce qui peut entraîner des délais d'expiration des transactions ou des conflits de contrôle optimiste de la concurrence (OCC).

Pour plus d'informations sur l'OCC, consultezModèle de mise QLDB simultansimultansimultansimultansimultansimultansimultansimultansimultanéité.

Transactions créées implicitement

La méthode QldbDriver.executeLambda accepte une fonction lambda qui reçoit une instance de TransactionExecutor, que vous pouvez utiliser pour exécuter des instructions. L'instance de`TransactionExecutor` encapsule une transaction créée implicitement.

Vous pouvez exécuter des instructions dans la fonction Lambda à l'aide de la méthode d'exécution de l'exécuteur de transactions. Le pilote valide implicitement la transaction lorsque la fonction Lambda est renvoyée.

> **Note**
>
> La`execute` méthode prend en charge à la fois les types Amazon Ion et les types natifs Node.js. Si vous transmettez un type natif Node.js comme argument à`execute`, le pilote le convertit en type Ion à l'aide`ion-js` du package (à condition que la conversion pour le type de données Node.js donné soit prise en charge). Pour les types de données et les règles de conversion pris en charge, consultez le fichier README Ion JavaScript DOM.

Les sections suivantes montrent comment exécuter des opérations CRUD de base, spécifier une logique de nouvelle tentative personnalisée et implémenter des contraintes d'unicité.

Table des matières

Création de tables

JavaScript

```
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        await txn.execute("CREATE TABLE Person");
    });
})();
```

TypeScript

```
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        await txn.execute('CREATE TABLE Person');
    });
}());
```

Création d'index

JavaScript

```
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        await txn.execute("CREATE INDEX ON Person (GovId)");
    });
})();
```

TypeScript

```
(async function(): Promise<void> {
```

```
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        await txn.execute('CREATE INDEX ON Person (GovId)');
    });
}());
```

Lecture de documents

JavaScript

```
(async function() {
    // Assumes that Person table has documents as follows:
    // { "GovId": "TOYENC486FH", "FirstName": "Brent" }
    await qldbDriver.executeLambda(async (txn) => {
        const results = (await txn.execute("SELECT * FROM Person WHERE GovId =
 'TOYENC486FH'")).getResultList();
        for (let result of results) {
            console.log(result.get('GovId')); // prints [String: 'TOYENC486FH']
            console.log(result.get('FirstName')); // prints [String: 'Brent']
        }
    });
}());
```

TypeScript

```
(async function(): Promise<void> {
    // Assumes that Person table has documents as follows:
    // { "GovId": "TOYENC486FH", "FirstName": "Brent" }
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        const results: dom.Value[] = (await txn.execute("SELECT * FROM Person WHERE
 GovId = 'TOYENC486FH'")).getResultList();
        for (let result of results) {
            console.log(result.get('GovId')); // prints [String: 'TOYENC486FH']
            console.log(result.get('FirstName')); // prints [String: 'Brent']
        }
    });
}());
```

Utilisation de paramètres de requête

L'exemple de code suivant utilise un paramètre de requête de type natif.

JavaScript

```
(async function() {
    // Assumes that Person table has documents as follows:
    // { "GovId": "TOYENC486FH", "FirstName": "Brent" }
    await qldbDriver.executeLambda(async (txn) => {
        const results = (await txn.execute('SELECT * FROM Person WHERE GovId = ?',
 'TOYENC486FH')).getResultList();
        for (let result of results) {
            console.log(result.get('GovId')); // prints [String: 'TOYENC486FH']
            console.log(result.get('FirstName')); // prints [String: 'Brent']
        }
    });
}());
```

TypeScript

```
(async function(): Promise<void> {
    // Assumes that Person table has documents as follows:
    // { "GovId": "TOYENC486FH", "FirstName": "Brent" }
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        const results: dom.Value[] = (await txn.execute('SELECT * FROM Person WHERE
 GovId = ?', 'TOYENC486FH')).getResultList();
        for (let result of results) {
            console.log(result.get('GovId')); // prints [String: 'TOYENC486FH']
            console.log(result.get('FirstName')); // prints [String: 'Brent']
        }
    });
}());
```

L'exemple de code suivant utilise un paramètre de requête de type Ion.

JavaScript

```
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        const govId = ionjs.load("TOYENC486FH");

        const results = (await txn.execute('SELECT * FROM Person WHERE GovId = ?',
 govId)).getResultList();
        for (let result of results) {
            console.log(result.get('GovId')); // prints [String: 'TOYENC486FH']
```

```
            console.log(result.get('FirstName')); // prints [String: 'Brent']
        }
    });
}());
```

TypeScript

```
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        const govId: dom.Value = load("TOYENC486FH");

        const results: dom.Value[] = (await txn.execute('SELECT * FROM Person WHERE
 GovId = ?', govId)).getResultList();
        for (let result of results) {
            console.log(result.get('GovId')); // prints [String: 'TOYENC486FH']
            console.log(result.get('FirstName')); // prints [String: 'Brent']
        }
    });
}());
```

L'exemple de code suivant utilise plusieurs paramètres de requête.

JavaScript

```
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        const results = (await txn.execute('SELECT * FROM Person WHERE GovId = ? AND
 FirstName = ?', 'TOYENC486FH', 'Brent')).getResultList();
        for (let result of results) {
            console.log(result.get('GovId')); // prints [String: 'TOYENC486FH']
            console.log(result.get('FirstName')); // prints [String: 'Brent']
        }
    });
}());
```

TypeScript

```
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        const results: dom.Value[] = (await txn.execute('SELECT * FROM Person WHERE
 GovId = ? AND FirstName = ?', 'TOYENC486FH', 'Brent')).getResultList();
        for (let result of results) {
```

```
            console.log(result.get('GovId')); // prints [String: 'TOYENC486FH']
            console.log(result.get('FirstName')); // prints [String: 'Brent']
        }
    });
}());
```

L'exemple de code suivant utilise une liste de paramètres de requête.

JavaScript

```
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        const govIds = ['TOYENC486FH','LOGANB486CG','LEWISR261LL'];
        /*
        Assumes that Person table has documents as follows:
        { "GovId": "TOYENC486FH", "FirstName": "Brent" }
        { "GovId": "LOGANB486CG", "FirstName": "Brent" }
        { "GovId": "LEWISR261LL", "FirstName": "Raul" }
        */
        const results = (await txn.execute('SELECT * FROM Person WHERE GovId IN
 (?,?,?)', ...govIds)).getResultList();
        for (let result of results) {
            console.log(result.get('GovId'));
            console.log(result.get('FirstName'));
            /*
            prints:
            [String: 'TOYENC486FH']
            [String: 'Brent']
            [String: 'LOGANB486CG']
            [String: 'Brent']
            [String: 'LEWISR261LL']
            [String: 'Raul']
            */
        }
    });
}());
```

TypeScript

```
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        const govIds: string[] = ['TOYENC486FH','LOGANB486CG','LEWISR261LL'];
```

```
        /*
        Assumes that Person table has documents as follows:
        { "GovId": "TOYENC486FH", "FirstName": "Brent" }
        { "GovId": "LOGANB486CG", "FirstName": "Brent" }
        { "GovId": "LEWISR261LL", "FirstName": "Raul" }
        */
        const results: dom.Value[] = (await txn.execute('SELECT * FROM Person WHERE
 GovId IN (?,?,?)', ...govIds)).getResultList();
        for (let result of results) {
            console.log(result.get('GovId'));
            console.log(result.get('FirstName'));
            /*
            prints:
            [String: 'TOYENC486FH']
            [String: 'Brent']
            [String: 'LOGANB486CG']
            [String: 'Brent']
            [String: 'LEWISR261LL']
            [String: 'Raul']
            */
        }
    });
}());
```

> **Note**
>
> Lorsque vous exécutez une requête sans recherche indexée, elle appelle une analyse complète de la table. Dans cet exemple, nous recommandons de disposer d'un index sur le`GovId` terrain afin d'optimiser les performances. Si aucun index n'est activé`GovId`, les requêtes peuvent présenter une latence plus importante et peuvent également entraîner des exceptions de conflit OCC ou des délais de transaction.

Insertion de documents

L'exemple de code suivant insère des types de données natifs.

JavaScript

```
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
```

```
        // Check if doc with GovId:TOYENC486FH exists
        // This is critical to make this transaction idempotent
        const results = (await txn.execute('SELECT * FROM Person WHERE GovId = ?',
 'TOYENC486FH')).getResultList();
        // Insert the document after ensuring it doesn't already exist
        if (results.length == 0) {
            const doc = {
                'FirstName': 'Brent',
                'GovId': 'TOYENC486FH',
            };
            await txn.execute('INSERT INTO Person ?', doc);
        }
    });
}());
```

TypeScript

```
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        // Check if doc with GovId:TOYENC486FH exists
        // This is critical to make this transaction idempotent
        const results: dom.Value[] = (await txn.execute('SELECT * FROM Person WHERE
 GovId = ?', 'TOYENC486FH')).getResultList();
        // Insert the document after ensuring it doesn't already exist
        if (results.length == 0) {
            const doc: Record<string, string> = {
                'FirstName': 'Brent',
                'GovId': 'TOYENC486FH',
            };
            await txn.execute('INSERT INTO Person ?', doc);
        }
    });
}());
```

L'exemple de code suivant insère des types de données Ion.

JavaScript

```
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        // Check if doc with GovId:TOYENC486FH exists
        // This is critical to make this transaction idempotent
```

```
        const results = (await txn.execute('SELECT * FROM Person WHERE GovId = ?',
 'TOYENC486FH')).getResultList();
        // Insert the document after ensuring it doesn't already exist
        if (results.length == 0) {
            const doc = {
                'FirstName': 'Brent',
                'GovId': 'TOYENC486FH',
            };
            // Create a sample Ion doc
            const ionDoc = ionjs.load(ionjs.dumpBinary(doc));

            await txn.execute('INSERT INTO Person ?', ionDoc);
        }
    });
}());
```

TypeScript

```
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        // Check if doc with GovId:TOYENC486FH exists
        // This is critical to make this transaction idempotent
        const results: dom.Value[] = (await txn.execute('SELECT * FROM Person WHERE
 GovId = ?', 'TOYENC486FH')).getResultList();
        // Insert the document after ensuring it doesn't already exist
        if (results.length == 0) {
            const doc: Record<string, string> = {
                'FirstName': 'Brent',
                'GovId': 'TOYENC486FH',
            };
            // Create a sample Ion doc
            const ionDoc: dom.Value = load(dumpBinary(doc));

            await txn.execute('INSERT INTO Person ?', ionDoc);
        }
    });
}());
```

Cette transaction insère un document dans le`Person` tableau. Avant de l'insérer, il vérifie d'abord si le document existe déjà dans le tableau. Ce contrôle confère à la transaction un caractère idempotent. Même si vous exécutez cette transaction plusieurs fois, elle n'aura aucun effet secondaire imprévu.

> **Note**
>
> Dans cet exemple, nous recommandons de disposer d'un index sur le`GovId` terrain afin d'optimiser les performances. Si aucun index n'est activé`GovId`, les instructions peuvent présenter une latence plus importante et peuvent également entraîner des exceptions de conflit OCC ou des délais de transaction.

Insertion de plusieurs documents dans une seule déclaration

Pour insérer plusieurs documents à l'aide d'une seuleINSERT instruction, vous pouvez transmettre un paramètre de type list à l'instruction comme suit.

```
// people is a list
txn.execute("INSERT INTO People ?", people);
```

Vous ne placez pas l'espace réservé à la variable (?) entre crochets (`<<...>>`) lorsque vous transmettez une liste. Dans les instructions manuelles de PartiQL, les crochets doubles indiquent une collection non ordonnée appelée sac.

Mise à jour des documents

L'exemple de code suivant utilise des types de données natifs.

JavaScript

```
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        await txn.execute('UPDATE Person SET FirstName = ? WHERE GovId = ?', 'John',
 'TOYENC486FH');
    });
}());
```

TypeScript

```
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        await txn.execute('UPDATE Person SET FirstName = ? WHERE GovId = ?', 'John',
 'TOYENC486FH');
    });
```

```
}());
```

L'exemple de code suivant utilise les types de données Ion.

JavaScript

```
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        const firstName = ionjs.load("John");
        const govId = ionjs.load("TOYENC486FH");

        await txn.execute('UPDATE Person SET FirstName = ? WHERE GovId = ?',
 firstName, govId);
    });
}());
```

TypeScript

```
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        const firstName: dom.Value = load("John");
        const govId: dom.Value = load("TOYENC486FH");

        await txn.execute('UPDATE Person SET FirstName = ? WHERE GovId = ?',
 firstName, govId);
    });
}());
```

> **Note**
>
> Dans cet exemple, nous recommandons de disposer d'un index sur le`GovId` terrain afin d'optimiser les performances. Si aucun index n'est activé`GovId`, les instructions peuvent présenter une latence plus importante et peuvent également entraîner des exceptions de conflit OCC ou des délais de transaction.

Supprimer des documents

L'exemple de code suivant utilise des types de données natifs.

JavaScript

```
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        await txn.execute('DELETE FROM Person WHERE GovId = ?', 'TOYENC486FH');
    });
}());
```

TypeScript

```
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        await txn.execute('DELETE FROM Person WHERE GovId = ?', 'TOYENC486FH');
    });
}());
```

L'exemple de code suivant utilise les types de données Ion.

JavaScript

```
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        const govId = ionjs.load("TOYENC486FH");

        await txn.execute('DELETE FROM Person WHERE GovId = ?', govId);
    });
}());
```

TypeScript

```
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        const govId: dom.Value = load("TOYENC486FH");

        await txn.execute('DELETE FROM Person WHERE GovId = ?', govId);
    });
}());
```

> **Note**
>
> Dans cet exemple, nous recommandons de disposer d'un index sur le`GovId` terrain afin d'optimiser les performances. Si aucun index n'est activé`GovId`, les instructions peuvent présenter une latence plus importante et peuvent également entraîner des exceptions de conflit OCC ou des délais de transaction.

Exécution de plusieurs relevés dans une transaction

TypeScript

```
// This code snippet is intentionally trivial. In reality you wouldn't do this
 because you'd
// set your UPDATE to filter on vin and insured, and check if you updated something
 or not.
async function insureCar(driver: QldbDriver, vin: string): Promise<boolean> {

    return await driver.executeLambda(async (txn: TransactionExecutor) => {
        const results: dom.Value[] = (await txn.execute(
            "SELECT insured FROM Vehicles WHERE vin = ? AND insured = FALSE",
 vin)).getResultList();

        if (results.length > 0) {
            await txn.execute(
                "UPDATE Vehicles SET insured = TRUE WHERE vin = ?", vin);
            return true;
        }
        return false;
    });
};
```

Nouvelle tentative de logique

La`executeLambda` méthode du pilote possède un mécanisme de nouvelle tentative intégré qui réessaie la transaction si une exception pouvant être réessayée se produit (par exemple, des délais d'attente ou des conflits OCC). Le nombre maximum de nouvelles tentatives et la stratégie de mise en pause sont configurables.

La limite de tentatives par défaut est de4, et la stratégie d'annulation par défaut est defaultBackoffFunctionbasée sur des10 millisecondes. Vous pouvez définir la configuration des nouvelles tentatives par instance de pilote et également par transaction en utilisant une instance de RetryConfig.

L'exemple de code suivant spécifie une logique de nouvelle tentative avec une limite de tentatives personnalisée et une stratégie d'arrêt personnalisée pour une instance du pilote.

JavaScript

```
var qldb = require('amazon-qldb-driver-nodejs');

// Configuring retry limit to 2
const retryConfig = new qldb.RetryConfig(2);
const qldbDriver = new qldb.QldbDriver("test-ledger", undefined, undefined,
 retryConfig);

// Configuring a custom backoff which increases delay by 1s for each attempt.
const customBackoff = (retryAttempt, error, transactionId) => {
    return 1000 * retryAttempt;
};

const retryConfigCustomBackoff = new qldb.RetryConfig(2, customBackoff);
const qldbDriverCustomBackoff = new qldb.QldbDriver("test-ledger", undefined,
 undefined, retryConfigCustomBackoff);
```

TypeScript

```
import { BackoffFunction, QldbDriver, RetryConfig } from "amazon-qldb-driver-nodejs"

// Configuring retry limit to 2
const retryConfig: RetryConfig = new RetryConfig(2);
const qldbDriver: QldbDriver = new QldbDriver("test-ledger", undefined, undefined,
 retryConfig);

// Configuring a custom backoff which increases delay by 1s for each attempt.
const customBackoff: BackoffFunction = (retryAttempt: number, error: Error,
 transactionId: string) => {
    return 1000 * retryAttempt;
};

const retryConfigCustomBackoff: RetryConfig = new RetryConfig(2, customBackoff);
```

```
const qldbDriverCustomBackoff: QldbDriver = new QldbDriver("test-ledger", undefined,
 undefined, retryConfigCustomBackoff);
```

L'exemple de code suivant spécifie une logique de nouvelle tentative avec une limite de tentatives personnalisée et une stratégie d'arrêt personnalisée pour une exécution Lambda particulière. Cette configuration`executeLambda` remplace la logique de nouvelle tentative définie pour l'instance du pilote.

JavaScript

```
var qldb = require('amazon-qldb-driver-nodejs');

// Configuring retry limit to 2
const retryConfig1 = new qldb.RetryConfig(2);
const qldbDriver = new qldb.QldbDriver("test-ledger", undefined, undefined,
 retryConfig1);

// Configuring a custom backoff which increases delay by 1s for each attempt.
const customBackoff = (retryAttempt, error, transactionId) => {
    return 1000 * retryAttempt;
};

const retryConfig2 = new qldb.RetryConfig(2, customBackoff);

// The config `retryConfig1` will be overridden by `retryConfig2`
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        await txn.execute('CREATE TABLE Person');
    }, retryConfig2);
}());
```

TypeScript

```
import { BackoffFunction, QldbDriver, RetryConfig, TransactionExecutor } from
 "amazon-qldb-driver-nodejs"

// Configuring retry limit to 2
const retryConfig1: RetryConfig = new RetryConfig(2);
const qldbDriver: QldbDriver = new QldbDriver("test-ledger", undefined, undefined,
 retryConfig1);
```

```
// Configuring a custom backoff which increases delay by 1s for each attempt.
const customBackoff: BackoffFunction = (retryAttempt: number, error: Error,
 transactionId: string) => {
    return 1000 * retryAttempt;
};

const retryConfig2: RetryConfig = new RetryConfig(2, customBackoff);

// The config `retryConfig1` will be overridden by `retryConfig2`
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        await txn.execute('CREATE TABLE Person');
    }, retryConfig2);
}());
```

Mise en œuvre de contraintes d'unicité

QLDB ne prend pas en charge les index uniques, mais vous pouvez implémenter ce comportement dans votre application.

Supposons que vous souhaitiez implémenter une contrainte d'unicité sur le`GovId` champ de la`Person` table. Pour ce faire, vous pouvez écrire une transaction qui effectue les actions suivantes :

1. Affirmez que la table ne contient aucun document existant portant une valeur spécifié`GovId`.
2. Insérez le document si l'assertion est acceptée.

Si une transaction concurrente passe simultanément l'assertion, seule l'une des transactions sera validée avec succès. L'autre transaction échouera avec une exception de conflit OCC.

L'exemple de code suivant montre comment mettre en œuvre cette logique de contrainte d'unicité.

JavaScript

```
const govId = 'TOYENC486FH';
const document = {
    'FirstName': 'Brent',
    'GovId': 'TOYENC486FH',
};
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        // Check if doc with GovId = govId exists
```

```
        const results = (await txn.execute('SELECT * FROM Person WHERE GovId = ?',
 govId)).getResultList();
        // Insert the document after ensuring it doesn't already exist
        if (results.length == 0) {
            await txn.execute('INSERT INTO Person ?', document);
        }
    });
})();
```

TypeScript

```
const govId: string = 'TOYENC486FH';
const document: Record<string, string> = {
    'FirstName': 'Brent',
    'GovId': 'TOYENC486FH',
};
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        // Check if doc with GovId = govId exists
        const results: dom.Value[] = (await txn.execute('SELECT * FROM Person WHERE
 GovId = ?', govId)).getResultList();
        // Insert the document after ensuring it doesn't already exist
        if (results.length == 0) {
            await txn.execute('INSERT INTO Person ?', document);
        }
    });
})();
```

> **Note**
>
> Dans cet exemple, nous recommandons de disposer d'un index sur le`GovId` terrain afin d'optimiser les performances. Si aucun index n'est activé`GovId`, les instructions peuvent présenter une latence plus importante et peuvent également entraîner des exceptions de conflit OCC ou des délais de transaction.

Utilisation d'Amazon Ion

Les sections suivantes montrent comment utiliser le module Amazon Ion pour traiter les données Ion.

Table des matières

Importation du module Ion

JavaScript

```
var ionjs = require('ion-js');
```

TypeScript

```
import { dom, dumpBinary, dumpText, load } from "ion-js";
```

Création de types Ion

L'exemple de code suivant crée un objet Ion à partir du texte Ion.

JavaScript

```
const ionText  = '{GovId: "TOYENC486FH", FirstName: "Brent"}';
const ionObj = ionjs.load(ionText);

console.log(ionObj.get('GovId')); // prints [String: 'TOYENC486FH']
console.log(ionObj.get('FirstName')); // prints [String: 'Brent']
```

TypeScript

```
const ionText: string = '{GovId: "TOYENC486FH", FirstName: "Brent"}';
const ionObj: dom.Value = load(ionText);

console.log(ionObj.get('GovId')); // prints [String: 'TOYENC486FH']
console.log(ionObj.get('FirstName')); // prints [String: 'Brent']
```

L'exemple de code suivant crée un objet Ion à partir d'un dictionnaire Node.js.

JavaScript

```
const aDict = {
    'GovId': 'TOYENC486FH',
    'FirstName': 'Brent'
};
const ionObj = ionjs.load(ionjs.dumpBinary(aDict));
console.log(ionObj.get('GovId')); // prints [String: 'TOYENC486FH']
console.log(ionObj.get('FirstName')); // prints [String: 'Brent']
```

TypeScript

```
const aDict: Record<string, string> = {
    'GovId': 'TOYENC486FH',
    'FirstName': 'Brent'
};
const ionObj: dom.Value = load(dumpBinary(aDict));
console.log(ionObj.get('GovId')); // prints [String: 'TOYENC486FH']
console.log(ionObj.get('FirstName')); // prints [String: 'Brent']
```

Obtenir un dump binaire Ion

JavaScript

```
// ionObj is an Ion struct
console.log(ionjs.dumpBinary(ionObj).toString()); // prints
 224,1,0,234,238,151,129,131,222,147,135,190,144,133,71,111,118,73,100,137,70,105,114,115,11
```

TypeScript

```
// ionObj is an Ion struct
console.log(dumpBinary(ionObj).toString()); // prints
 224,1,0,234,238,151,129,131,222,147,135,190,144,133,71,111,118,73,100,137,70,105,114,115,11
```

Obtenir un dump de texte Ion

JavaScript

```
// ionObj is an Ion struct
```

```
console.log(ionjs.dumpText(ionObj)); // prints
 {GovId:"TOYENC486FH",FirstName:"Brent"}
```

TypeScript

```
// ionObj is an Ion struct
console.log(dumpText(ionObj)); // prints {GovId:"TOYENC486FH",FirstName:"Brent"}
```

Pour plus d'informations sur Ion, consultez la documentation Amazon Ion sur GitHub. Pour plus d'exemples de code illustrant l'utilisation d'Ion dans QLDB, consultezUtilisation des types de données Amazon Ion dans Amazon QLDB.

Pilote Amazon QLDB pour Python

Pour utiliser les données de votre registre, vous pouvez vous connecter à Amazon QLDB depuis votre application Python à l'aide d'un piloteAWS fourni. Les rubriques suivantes décrivent comment démarrer avec le pilote QLDB pour Python.

Rubriques

- Ressources du conducteur
- Prérequis
- Installation
- Pilote Amazon QLDB pour Python — Tutoriel de démarrage rapide
- Pilote Amazon QLDB pour Python — Référence du livre de recettes

Ressources du conducteur

Pour plus d'informations sur les fonctionnalités prises en charge par le pilote Python, consultez les ressources suivantes :

- Référence de l'API : 3.x, 2.x
- Code source du pilote (GitHub)
- Exemple de code source d'application (GitHub)
- Exemples de code Amazon Ion

Prérequis

Avant de commencer à installer le pilote QLDB pour Python, vous devez effectuer les opérations suivantes :

1. Suivez les instructions deAWS configuration dansAccès à Amazon QLDB. Cela inclut les éléments suivants :

   1. S'inscrire àAWS.
   2. Créez un utilisateur avec les autorisations QLDB appropriées.
   3. Accorder un accès par programmation à des fins de développement.
2. Installez l'une des versions suivantes de Python à partir du site de téléchargement de Python :

   - 3.6 ou version ultérieure — pilote QLDB pour Python v3
   - 3.4 ou version ultérieure — pilote QLDB pour Python v2
3. Configurez vosAWS informations d'identification et celles par défautRégion AWS. Pour obtenir des instructions, voir Démarrage rapide dans laAWS SDK for Python (Boto3) documentation.

   Pour obtenir la liste complète des régions disponibles, consultez les points de terminaison et les quotas Amazon QLDB dans le Références générales AWS.

Vous pouvez ensuite télécharger l'exemple d'application complet du didacticiel, ou vous pouvez uniquement installer le pilote dans un projet Python et exécuter des exemples de code courts.

- Pour installer le pilote QLDB et leAWS SDK for Python (Boto3) dans un projet existant, passez àInstallation.
- Pour configurer un projet et exécuter des exemples de code abrégé illustrant les transactions de données de base sur un registre, consultez leDidacticiel de démarrage rapide.
- Pour exécuter des exemples plus détaillés d'opérations sur les données et les API de gestion dans l'exemple d'application complet du didacticiel, consultez leTutoriel Python.

Installation

QLDB prend en charge les versions de pilotes suivantes et leurs dépendances Python.

Versions du pilote	Version Python	État	Dernière date de parution
2.x	3.4 ou version ultérieure	Version de production	7 mai 2020
3.x	3.6 ou version ultérieure	Version de production	28 octobre 2021

Pour installer le pilote QLDB à partir de PyPI à l'aide de `pip` (un gestionnaire de packages pour Python), entrez ce qui suit sur la ligne de commande.

3.x

```
pip install pyqldb
```

2.x

```
pip install pyqldb==2.0.2
```

L'installation du pilote entraîne également l'installation de ses dépendances, notamment les packages Amazon Ion AWS SDK for Python (Boto3)et Amazon.

Utilisation du pilote pour se connecter à un registre

Vous pouvez ensuite importer le pilote et l'utiliser pour vous connecter à un registre. L'exemple de code Python suivant montre comment créer une session pour un nom de registre spécifié.

3.x

```
from pyqldb.driver.qldb_driver import QldbDriver
qldb_driver = QldbDriver(ledger_name='testLedger')

for table in qldb_driver.list_tables():
    print(table)
```

2.x

```
from pyqldb.driver.pooled_qldb_driver import PooledQldbDriver
```

```
qldb_driver = PooledQldbDriver(ledger_name='testLedger')
qldb_session = qldb_driver.get_session()

for table in qldb_session.list_tables():
    print(table)
```

Pour des exemples de code courts expliquant comment exécuter des transactions de données de base sur un registre, consultez leRéférence de livre de recettes.

Pilote Amazon QLDB pour Python — Tutoriel de démarrage rapide

Dans ce didacticiel, vous apprendrez comment configurer une application simple à l'aide de la dernière version du pilote Amazon QLDB pour Python. Ce guide comprend les étapes d'installation du pilote et des exemples de code court pour des opérations de création, lecture, mise à jour et suppression (CRUD). Pour des exemples plus détaillés illustrant ces opérations dans un exemple d'application complet, consultez leTutoriel Python.

Rubriques

Prérequis

Avant de commencer, veillez à exécuter les actions suivantes :

1. Complétez lePrérequis pour le pilote Python, si vous ne l'avez pas déjà fait. Cela inclut l'inscriptionAWS, l'octroi d'un accès programmatique pour le développement et installation de Python version 3.6 ou ultérieure.

2. Créez un registre nomméquick-start.

   Pour savoir comment créer un registre, consultezOpérations de base pour les registres Amazon QLDB ouÉtape 1 : Créer un nouveau registre dans Prise en main de la console.

Étape 1 : Configurer votre projet

Tout d'abord, configurez votre projet Python.

> **Note**
>
> Si vous utilisez un IDE doté de fonctionnalités permettant d'automatiser ces étapes de configuration, vous pouvez passer directement àÉtape 2 : Initialiser le pilote.

1. Créez un dossier pour votre application.

   ```
   $ mkdir myproject
   $ cd myproject
   ```

2. Pour installer le pilote QLDB pour Python à partir de PyPI, entrez lapip commande suivante.

   ```
   $ pip install pyqldb
   ```

   L'installation du pilote installe également ses dépendances, notamment les packages Amazon Ion AWS SDK for Python (Boto3)et Amazon.

3. Créez un fichier nommé app.py.

   Ajoutez ensuite progressivement les exemples de code dans les étapes suivantes pour essayer certaines opérations CRUD de base. Vous pouvez également ignorer le step-by-step didacticiel et exécuter l'application complète.

Étape 2 : Initialiser le pilote

Initialisez une instance du pilote qui se connecte au registre nomméquick-start. Ajoutez le code suivant à votreapp.py fichier.

```
from pyqldb.config.retry_config import RetryConfig
```

```
from pyqldb.driver.qldb_driver import QldbDriver

# Configure retry limit to 3
retry_config = RetryConfig(retry_limit=3)

# Initialize the driver
print("Initializing the driver")
qldb_driver = QldbDriver("quick-start", retry_config=retry_config)
```

Étape 3 : création d'une table et index

L'exemple de code suivant montre comment exécuter`CREATE TABLE` des`CREATE INDEX` instructions.

Ajoutez le code suivant qui crée une table nommée`People` et un index pour le`lastName` champ de cette table. Les index sont nécessaires pour optimiser les performances des requêtes et contribuer à limiter les exceptions de conflit liées au contrôle optimiste de la concurrence (OCC).

```
def create_table(transaction_executor):
    print("Creating a table")
    transaction_executor.execute_statement("Create TABLE People")

def create_index(transaction_executor):
    print("Creating an index")
    transaction_executor.execute_statement("CREATE INDEX ON People(lastName)")

# Create a table
qldb_driver.execute_lambda(lambda executor: create_table(executor))

# Create an index on the table
qldb_driver.execute_lambda(lambda executor: create_index(executor))
```

Étape 4 : Insérer un document

L'exemple de code suivant montre comment exécuter une`INSERT` instruction. QLDB prend en charge le langage de requête PartiQL (compatible SQL) et le format de données Amazon Ion (surensemble de JSON).

Ajoutez le code suivant pour insérer un document dans le`People` tableau.

```
def insert_documents(transaction_executor, arg_1):
```

```
    print("Inserting a document")
    transaction_executor.execute_statement("INSERT INTO People ?", arg_1)

# Insert a document
doc_1 = { 'firstName': "John",
          'lastName': "Doe",
          'age': 32,
        }

qldb_driver.execute_lambda(lambda x: insert_documents(x, doc_1))
```

Cet exemple utilise un point d'interrogation (?) comme espace réservé variable pour transmettre les informations du document à l'instruction. La`execute_statement` méthode prend en charge les valeurs des types Amazon Ion et des types natifs de Python.

> **Tip**
>
> Pour insérer plusieurs documents à l'aide d'une seuleINSERT instruction, vous pouvez transmettre un paramètre de type list à l'instruction comme suit.
>
> ```
> # people is a list
> transaction_executor.execute_statement("INSERT INTO Person ?", people)
> ```
>
> Vous ne placez pas l'espace réservé à la variable (?) entre crochets (<<...>>) lorsque vous transmettez une liste. Dans les instructions manuelles de PartiQL, les crochets doubles indiquent une collection non ordonnée appelée sac.

Étape 5 : interroger le document

L'exemple de code suivant montre comment exécuter une`SELECT` instruction.

Ajoutez le code suivant qui interroge un document à partir du`People` tableau.

```
def read_documents(transaction_executor):
    print("Querying the table")
    cursor = transaction_executor.execute_statement("SELECT * FROM People WHERE
 lastName = ?", 'Doe')

    for doc in cursor:
```

```
        print(doc["firstName"])
        print(doc["lastName"])
        print(doc["age"])

# Query the table
qldb_driver.execute_lambda(lambda executor: read_documents(executor))
```

Étape 6 : Mettre à jour le document

L'exemple de code suivant montre comment exécuter uneUPDATE instruction.

1. Ajoutez le code suivant pour mettre à jour un document dans lePeople tableau en le mettant à jourage vers42.

   ```
   def update_documents(transaction_executor, age, lastName):
       print("Updating the document")
       transaction_executor.execute_statement("UPDATE People SET age = ? WHERE
    lastName = ?", age, lastName)

   # Update the document
   age = 42
   lastName = 'Doe'

   qldb_driver.execute_lambda(lambda x: update_documents(x, age, lastName))
   ```

2. Interrogez à nouveau la table pour voir la valeur mise à jour.

   ```
   # Query the updated document
   qldb_driver.execute_lambda(lambda executor: read_documents(executor))
   ```

3. Pour exécuter l'application, saisissez la commande suivante dans le répertoire de votre projet.

   ```
   $ python app.py
   ```

Exécution de l'application complète

L'exemple de code suivant représente la version complète de l'app.pyapplication. Au lieu de suivre les étapes précédentes individuellement, vous pouvez également copier et exécuter cet exemple de code du début à la fin. Cette application montre certaines opérations CRUD de base sur le registre nomméquick-start.

> **Note**
>
> Avant d'exécuter ce code, assurez-vous qu'aucune table active n'est déjà nomméePeople dans lequick-start registre.

```
from pyqldb.config.retry_config import RetryConfig
from pyqldb.driver.qldb_driver import QldbDriver

def create_table(transaction_executor):
    print("Creating a table")
    transaction_executor.execute_statement("CREATE TABLE People")

def create_index(transaction_executor):
    print("Creating an index")
    transaction_executor.execute_statement("CREATE INDEX ON People(lastName)")

def insert_documents(transaction_executor, arg_1):
    print("Inserting a document")
    transaction_executor.execute_statement("INSERT INTO People ?", arg_1)

def read_documents(transaction_executor):
    print("Querying the table")
    cursor = transaction_executor.execute_statement("SELECT * FROM People WHERE
 lastName = ?", 'Doe')


    for doc in cursor:
        print(doc["firstName"])
        print(doc["lastName"])
        print(doc["age"])

def update_documents(transaction_executor, age, lastName):
    print("Updating the document")
    transaction_executor.execute_statement("UPDATE People SET age = ? WHERE lastName
 = ?", age, lastName)

# Configure retry limit to 3
retry_config = RetryConfig(retry_limit=3)

# Initialize the driver
print("Initializing the driver")
```

```
qldb_driver = QldbDriver("quick-start", retry_config=retry_config)

# Create a table
qldb_driver.execute_lambda(lambda executor: create_table(executor))

# Create an index on the table
qldb_driver.execute_lambda(lambda executor: create_index(executor))

# Insert a document
doc_1 = { 'firstName': "John",
          'lastName': "Doe",
          'age': 32,
        }

qldb_driver.execute_lambda(lambda x: insert_documents(x, doc_1))

# Query the table
qldb_driver.execute_lambda(lambda executor: read_documents(executor))

# Update the document
age = 42
lastName = 'Doe'

qldb_driver.execute_lambda(lambda x: update_documents(x, age, lastName))

# Query the table for the updated document
qldb_driver.execute_lambda(lambda executor: read_documents(executor))
```

Pour exécuter l'application complète, saisissez la commande suivante dans le répertoire de votre projet.

```
$ python app.py
```

Pilote Amazon QLDB pour Python — Référence du livre de recettes

Ce guide de référence présente des cas d'utilisation courants du pilote Amazon QLDB pour Python. Il fournit des exemples de code Python montrant comment utiliser le pilote pour exécuter des opérations de création, lecture, mise à jour et suppression (CRUD) typiques. Il inclut également des exemples de code pour le traitement des données Amazon Ion. En outre, ce guide met en évidence les meilleures pratiques pour rendre les transactions idempotentes et mettre en œuvre des contraintes d'unicité.

> **Note**
>
> Le cas échéant, certains cas d'utilisation comportent des exemples de code différents pour chaque version majeure prise en charge du pilote QLDB pour Python.

Table des matières

Importation du pilote

L'exemple de code suivant importe le pilote.

3.x

```
from pyqldb.driver.qldb_driver import QldbDriver
import amazon.ion.simpleion as simpleion
```

2.x

```
from pyqldb.driver.pooled_qldb_driver import PooledQldbDriver
import amazon.ion.simpleion as simpleion
```

> **Note**
>
> Cet exemple importe également le package Amazon Ion (`amazon.ion.simpleion`). Vous avez besoin de ce package pour traiter les données Ion lors de l'exécution de certaines opérations de données dans cette référence. Pour en savoir plus, consultez Utilisation d'Amazon Ion.

Instanciation du pilote

L'exemple de code suivant crée une instance du pilote qui se connecte à un nom de registre spécifié à l'aide des paramètres par défaut.

3.x

```
qldb_driver = QldbDriver(ledger_name='vehicle-registration')
```

2.x

```
qldb_driver = PooledQldbDriver(ledger_name='vehicle-registration')
```

Opérations CRUD

QLDB exécute des opérations de création, lecture, mise à jour et suppression (CRUD) dans le cadre d'une transaction.

> **Warning**
>
> À titre de bonne pratique, faites en sorte que vos transactions d'écriture soient strictement idempotentes.

Rendre les transactions idempotentes

Nous vous recommandons de rendre les transactions d'écriture idempotentes afin d'éviter tout effet secondaire inattendu en cas de nouvelles tentatives. Une transaction est idempotente si elle peut être exécutée plusieurs fois et produire des résultats identiques à chaque fois.

Prenons l'exemple d'une transaction qui insère un document dans une table nomméePerson. La transaction doit d'abord vérifier si le document existe déjà dans le tableau. Sans cette vérification, le tableau risque de contenir des documents dupliqués.

Supposons que QLDB valide avec succès la transaction côté serveur, mais que le client expire en attendant une réponse. Si la transaction n'est pas idempotente, le même document peut être inséré plusieurs fois en cas de nouvelle tentative.

Utilisation d'index pour éviter l'analyse complète des tables

Nous vous recommandons également d'exécuter des instructions avec une clause deWHERE prédicat à l'aide d'un opérateur d'égalité sur un champ indexé ou un identifiant de document, par exemple,`WHERE indexedField = 123` ou`WHERE indexedField IN (456, 789)`. Sans cette recherche indexée, QLDB doit effectuer une analyse des tables, ce qui peut entraîner des délais d'expiration des transactions ou des conflits de contrôle optimiste de la concurrence (OCC).

Pour plus d'informations sur l'OCC, consultezModèle de mise QLDB simultansimultansimultansimultansimultansimultansimultansimultansimultanéité.

Transactions créées implicitement

La méthode pyqldb.driver.qldb_driver.execute_lambda accepte une fonction lambda qui reçoit une instance de PyQLDB.Execution.Executor.Executor, que vous pouvez utiliser pour exécuter des instructions. L'instance de`Executor` encapsule une transaction créée implicitement.

Vous pouvez exécuter des instructions dans la fonction lambda en utilisant la méthode execute_statement de l'exécuteur de transactions. Le pilote valide implicitement la transaction lorsque la fonction Lambda est renvoyée.

> **Note**
>
> La`execute_statement` méthode prend en charge à la fois les types Amazon Ion et les types natifs de Python. Si vous transmettez un type natif Python comme argument à`execute_statement`, le pilote le convertit en type Ion à l'aide du`amazon.ion.simpleion` module (à condition que la conversion pour le type de données Python donné soit prise en charge). Pour connaître les types de données et les règles de conversion pris en charge, consultez le code source de Simpleion.

Les sections suivantes montrent comment exécuter des opérations CRUD de base, spécifier une logique de nouvelle tentative personnalisée et implémenter des contraintes d'unicité.

Table des matières

Création de tables

```
def create_table(transaction_executor):
    transaction_executor.execute_statement("CREATE TABLE Person")

qldb_driver.execute_lambda(lambda executor: create_table(executor))
```

Création d'index

```
def create_index(transaction_executor):
    transaction_executor.execute_statement("CREATE INDEX ON Person(GovId)")

qldb_driver.execute_lambda(lambda executor: create_index(executor))
```

Lire des documents

```
# Assumes that Person table has documents as follows:
# { "GovId": "TOYENC486FH", "FirstName": "Brent" }

def read_documents(transaction_executor):
    cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE GovId =
 'TOYENC486FH'")

    for doc in cursor:
        print(doc["GovId"]) # prints TOYENC486FH
        print(doc["FirstName"]) # prints Brent

qldb_driver.execute_lambda(lambda executor: read_documents(executor))
```

Utilisation des paramètres de la requête

L'exemple de code suivant utilise un paramètre de requête de type natif.

```
cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE GovId = ?",
 'TOYENC486FH')
```

L'exemple de code suivant utilise un paramètre de requête de type Ion.

```
name = ion.loads('Brent')
cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE FirstName
 = ?", name)
```

L'exemple de code suivant utilise plusieurs paramètres de requête.

```
cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE GovId = ?
 AND FirstName = ?", 'TOYENC486FH', "Brent")
```

L'exemple de code suivant utilise une liste de paramètres de requête.

```
gov_ids = ['TOYENC486FH','ROEE1','YH844']
cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE GovId IN
 (?,?,?)", *gov_ids)
```

> **Note**
>
> Lorsque vous exécutez une requête sans recherche indexée, elle appelle une analyse complète de la table. Dans cet exemple, nous recommandons de disposer d'un index sur le`GovId` terrain afin d'optimiser les performances. Si aucun index n'est activé`GovId`, les requêtes peuvent présenter une latence plus importante et peuvent également entraîner des exceptions de conflit OCC ou des délais de transaction.

Insertion de documents

L'exemple de code suivant insère des types de données natifs.

```
def insert_documents(transaction_executor, arg_1):
    # Check if doc with GovId:TOYENC486FH exists
    # This is critical to make this transaction idempotent
    cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE GovId
 = ?", 'TOYENC486FH')
    # Check if there is any record in the cursor
    first_record = next(cursor, None)

    if first_record:
        # Record already exists, no need to insert
        pass
    else:
        transaction_executor.execute_statement("INSERT INTO Person ?", arg_1)

doc_1 = { 'FirstName': "Brent",
          'GovId': 'TOYENC486FH',
        }

qldb_driver.execute_lambda(lambda executor: insert_documents(executor, doc_1))
```

L'exemple de code suivant insère des types de données Ion.

```
def insert_documents(transaction_executor, arg_1):
    # Check if doc with GovId:TOYENC486FH exists
```

```
    # This is critical to make this transaction idempotent
    cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE GovId
 = ?", 'TOYENC486FH')
    # Check if there is any record in the cursor
    first_record = next(cursor, None)

    if first_record:
        # Record already exists, no need to insert
        pass
    else:
        transaction_executor.execute_statement("INSERT INTO Person ?", arg_1)

doc_1 = { 'FirstName': 'Brent',
          'GovId': 'TOYENC486FH',
        }

# create a sample Ion doc
ion_doc_1 = simpleion.loads(simpleion.dumps(doc_1))

qldb_driver.execute_lambda(lambda executor: insert_documents(executor, ion_doc_1))
```

Cette transaction insère un document dans le`Person` tableau. Avant de l'insérer, il vérifie d'abord si le document existe déjà dans le tableau. Ce contrôle confère à la transaction un caractère idempotent. Même si vous exécutez cette transaction plusieurs fois, elle n'aura aucun effet secondaire imprévu.

> **Note**
>
> Dans cet exemple, nous recommandons de disposer d'un index sur le`GovId` terrain afin d'optimiser les performances. Si aucun index n'est activé`GovId`, les instructions peuvent présenter une latence plus importante et peuvent également entraîner des exceptions de conflit OCC ou des délais de transaction.

Insertion de plusieurs documents dans une seule déclaration

Pour insérer plusieurs documents à l'aide d'une seuleINSERT instruction, vous pouvez transmettre un paramètre de type list à l'instruction comme suit.

```
# people is a list
transaction_executor.execute_statement("INSERT INTO Person ?", people)
```

Vous ne placez pas l'espace réservé à la variable (?) entre crochets (<<...>>) lorsque vous transmettez une liste. Dans les instructions manuelles de PartiQL, les crochets doubles indiquent une collection non ordonnée appelée sac.

Mise à jour des documents

L'exemple de code suivant utilise des types de données natifs.

```
def update_documents(transaction_executor, gov_id, name):
    transaction_executor.execute_statement("UPDATE Person SET FirstName = ?  WHERE
 GovId = ?", name, gov_id)

gov_id = 'TOYENC486FH'
name = 'John'

qldb_driver.execute_lambda(lambda executor: update_documents(executor, gov_id, name))
```

L'exemple de code suivant utilise les types de données Ion.

```
def update_documents(transaction_executor, gov_id, name):
    transaction_executor.execute_statement("UPDATE Person SET FirstName = ? WHERE GovId
 = ?", name, gov_id)

# Ion datatypes
gov_id = simpleion.loads('TOYENC486FH')
name = simpleion.loads('John')

qldb_driver.execute_lambda(lambda executor: update_documents(executor, gov_id, name))
```

> **Note**
>
> Dans cet exemple, nous recommandons de disposer d'un index sur le`GovId` terrain afin d'optimiser les performances. Si aucun index n'est activé`GovId`, les instructions peuvent présenter une latence plus importante et peuvent également entraîner des exceptions de conflit OCC ou des délais de transaction.

Supprimer des documents

L'exemple de code suivant utilise des types de données natifs.

```
def delete_documents(transaction_executor, gov_id):
    cursor = transaction_executor.execute_statement("DELETE FROM Person WHERE GovId
 = ?", gov_id)

gov_id = 'TOYENC486FH'

qldb_driver.execute_lambda(lambda executor: delete_documents(executor, gov_id))
```

L'exemple de code suivant utilise les types de données Ion.

```
def delete_documents(transaction_executor, gov_id):
    cursor = transaction_executor.execute_statement("DELETE FROM Person WHERE GovId
 = ?", gov_id)

# Ion datatypes
gov_id = simpleion.loads('TOYENC486FH')

qldb_driver.execute_lambda(lambda executor: delete_documents(executor, gov_id))
```

> **Note**
>
> Dans cet exemple, nous recommandons de disposer d'un index sur le`GovId` terrain afin d'optimiser les performances. Si aucun index n'est activé`GovId`, les instructions peuvent présenter une latence plus importante et peuvent également entraîner des exceptions de conflit OCC ou des délais de transaction.

Exécution de plusieurs relevés dans une transaction

```
# This code snippet is intentionally trivial. In reality you wouldn't do this because
 you'd
# set your UPDATE to filter on vin and insured, and check if you updated something or
 not.

def do_insure_car(transaction_executor, vin):
    cursor = transaction_executor.execute_statement(
        "SELECT insured FROM Vehicles WHERE vin = ? AND insured = FALSE", vin)
    first_record = next(cursor, None)
    if first_record:
        transaction_executor.execute_statement(
            "UPDATE Vehicles SET insured = TRUE WHERE vin = ?", vin)
```

```
        return True
    else:
        return False

def insure_car(qldb_driver, vin_to_insure):
    return qldb_driver.execute_lambda(
        lambda executor: do_insure_car(executor, vin_to_insure))
```

Nouvelle tentative de nouvelle tentative

La`execute_lambda` méthode du pilote possède un mécanisme de nouvelle tentative intégré qui réessaie la transaction si une exception pouvant être réessayée se produit (par exemple, des délais d'attente ou des conflits OCC).

3.x

Le nombre maximum de nouvelles tentatives et la stratégie de nouvelle tentative sont configurables.

La limite de tentatives par défaut est de`4`, et la stratégie d'arrêt par défaut est Exponential Backoff and Jitter, avec une base de`10` millisecondes. Vous pouvez définir la configuration de nouvelle tentative par instance de pilote et également par transaction en utilisant une instance de pyqldb.config.retry_config. RetryConfig.

L'exemple de code suivant spécifie une logique de nouvelle tentative avec une limite de tentatives personnalisée et une stratégie d'arrêt personnalisée pour une instance du pilote.

```
from pyqldb.config.retry_config import RetryConfig
from pyqldb.driver.qldb_driver import QldbDriver

# Configuring retry limit to 2
retry_config = RetryConfig(retry_limit=2)
qldb_driver = QldbDriver("test-ledger", retry_config=retry_config)

# Configuring a custom backoff which increases delay by 1s for each attempt.
def custom_backoff(retry_attempt, error, transaction_id):
    return 1000 * retry_attempt

retry_config_custom_backoff = RetryConfig(retry_limit=2,
 custom_backoff=custom_backoff)
qldb_driver = QldbDriver("test-ledger", retry_config=retry_config_custom_backoff)
```

L'exemple de code suivant spécifie une logique de nouvelle tentative avec une limite de tentatives personnalisée et une stratégie d'arrêt personnalisée pour une exécution Lambda particulière. Cette configuration`execute_lambda` remplace la logique de nouvelle tentative définie pour l'instance du pilote.

```
from pyqldb.config.retry_config import RetryConfig
from pyqldb.driver.qldb_driver import QldbDriver

# Configuring retry limit to 2
retry_config_1 = RetryConfig(retry_limit=4)
qldb_driver = QldbDriver("test-ledger", retry_config=retry_config_1)

# Configuring a custom backoff which increases delay by 1s for each attempt.
def custom_backoff(retry_attempt, error, transaction_id):
    return 1000 * retry_attempt

retry_config_2 = RetryConfig(retry_limit=2, custom_backoff=custom_backoff)

# The config `retry_config_1` will be overriden by `retry_config_2`
qldb_driver.execute_lambda(lambda txn: txn.execute_statement("CREATE TABLE Person"),
 retry_config_2)
```

2.x

Le nombre maximum de nouvelles tentatives maximum est configurable. Vous pouvez configurer la limite de tentatives en définissant la`retry_limit` propriété lors de l'initialisation`PooledQldbDriver`.

La limite de tentatives par défaut est de`4`.

Mise en œuvre des contraintes d'unicité

QLDB ne prend pas en charge les index uniques, mais vous pouvez implémenter ce comportement dans votre application.

Supposons que vous souhaitiez implémenter une contrainte d'unicité sur le`GovId` champ de la`Person` table. Pour ce faire, vous pouvez écrire une transaction qui effectue les actions suivantes :

1. Affirmez que la table ne contient aucun document existant portant une valeur spécifiée`GovId`.
2. Insérez le document si l'assertion est acceptée.

Si une transaction concurrente passe simultanément l'assertion, seule l'une des transactions sera validée avec succès. L'autre transaction échouera avec une exception de conflit OCC.

L'exemple de code suivant montre comment mettre en œuvre cette logique de contrainte d'unicité.

```
def insert_documents(transaction_executor, gov_id, document):
    # Check if doc with GovId = gov_id exists
    cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE GovId
 = ?", gov_id)
    # Check if there is any record in the cursor
    first_record = next(cursor, None)

    if first_record:
        # Record already exists, no need to insert
        pass
    else:
        transaction_executor.execute_statement("INSERT INTO Person ?", document)

qldb_driver.execute_lambda(lambda executor: insert_documents(executor, gov_id,
 document))
```

> **Note**
>
> Dans cet exemple, nous recommandons de disposer d'un index sur le`GovId` terrain afin d'optimiser les performances. Si aucun index n'est activé`GovId`, les instructions peuvent présenter une latence plus importante et peuvent également entraîner des exceptions de conflit OCC ou des délais de transaction.

Utilisation d'Amazon Ion

Les sections suivantes montrent comment utiliser le module Amazon Ion pour traiter les données Ion.

Table des matières

Importation du module Ion

```
import amazon.ion.simpleion as simpleion
```

Création de types Ion

L'exemple de code suivant crée un objet Ion à partir du texte Ion.

```
ion_text = '{GovId: "TOYENC486FH", FirstName: "Brent"}'
ion_obj = simpleion.loads(ion_text)

print(ion_obj['GovId']) # prints TOYENC486FH
print(ion_obj['Name']) # prints Brent
```

L'exemple de code suivant crée un objet Ion à partir d'un Python`dict`.

```
a_dict = { 'GovId': 'TOYENC486FH',
           'FirstName': "Brent"
         }
ion_obj = simpleion.loads(simpleion.dumps(a_dict))

print(ion_obj['GovId']) # prints TOYENC486FH
print(ion_obj['FirstName']) # prints Brent
```

Obtenir un dump binaire Ion

```
# ion_obj is an Ion struct
print(simpleion.dumps(ion_obj)) # b'\xe0\x01\x00\xea\xee\x97\x81\x83\xde\x93\x87\xbe
\x90\x85GovId\x89FirstName\xde\x94\x8a\x8bTOYENC486FH\x8b\x85Brent'
```

Obtenir un dump de texte Ion

```
# ion_obj is an Ion struct
print(simpleion.dumps(ion_obj, binary=False)) # prints $ion_1_0
 {GovId:'TOYENC486FH',FirstName:"Brent"}
```

Pour plus d'informations sur l'utilisation d'Ion, consultez la documentation Amazon Ion sur GitHub. Pour plus d'exemples de code illustrant l'utilisation d'Ion dans QLDB, consultezUtilisation des types de données Amazon Ion dans Amazon QLDB.

Comprendre la gestion des sessions avec le pilote dans Amazon QLDB

Si vous avez une expérience concernant l'utilisation d'un système de gestion de base de données relationnelle (SGBDR), il se peut que vous ayez une connaissance concernant les connexions simultanées. QLDB n'a pas le même concept qu'une connexion RDBMS traditionnelle car les transactions sont exécutées avec des messages de requête et de réponse HTTP.

Dans QLDB, le concept analogue est une session active. Sur le plan conceptuel, une session est similaire à une connexion utilisateur : elle gère les informations relatives à vos demandes de transactions de données vers un registre. Une session active est une session qui exécute activement une transaction. Il peut également s'agir d'une session qui vient de terminer une transaction au cours de laquelle le service prévoit d'en démarrer une autre immédiatement. QLDB prend en charge une transaction active par session.

La limite du nombre de sessions actives simultanées par registre est définie dansQuotas et limites d'Amazon QLDB. Une fois cette limite atteinte, toute session qui tente de démarrer une transaction génère une erreur (`LimitExceededException`).

Pour connaître les meilleures pratiques relatives à la configuration d'un pool de sessions dans votre application à l'aide du pilote QLDB, consultezConfiguration de QldbDriver objet les recommandations relatives aux pilotes Amazon QLDB.

Table des matières

Cycle de vie de session

La séquence suivante des opérations de l'API de session QLDB représente le cycle de vie typique d'une session QLDB :

1. `StartSession`

2. `StartTransaction`
3. `ExecuteStatement`
4. `CommitTransaction`
5. Répétez les étapes 2 à 4 pour démarrer d'autres transactions (une transaction à la fois).
6. `EndSession`

Expiration de session

QLDB expire et annule une session après une durée de vie totale de 13 à 17 minutes, qu'elle soit ou non en cours d'exécution active d'une transaction. Les sessions peuvent être perdues ou altérées pour diverses raisons, telles qu'une panne matérielle, une défaillance du réseau ou le redémarrage d'applications. QLDB impose une durée de vie maximale aux sessions afin de garantir la résilience de l'application cliente en cas d'échec de session.

Gestion des sessions dans le pilote QLDB

Bien que vous puissiez utiliser unAWS SDK pour interagir directement avec l'API de session QLDB, cela ajoute de la complexité et vous oblige à calculer un résumé des validations. QLDB utilise ce condensé de validation pour garantir l'intégrité des transactions. Au lieu d'interagir directement avec cette API, nous vous recommandons d'utiliser le pilote QLDB.

Le pilote fournit une couche d'abstraction de haut niveau au-dessus de l'API de données transactionnelles. Il rationalise le processus d'exécution d'instructions partiQL sur les données du registre en gérant les appels SendCommandd'API. Ces appels d'API nécessitent plusieurs paramètres que le pilote gère pour vous, notamment la gestion des sessions, des transactions et la politique de relance en cas d'erreur.

Rubriques

- Présentation du regroupement de sessions
- Regroupement de sessions et logique transactionnelle
- Retour des sessions à la piscine

Présentation du regroupement de sessions

Dans les anciennes versions du pilote QLDB (par exemple, Java v1.1.0), nous avons fourni deux implémentations de l'objet pilote : une version standard, non groupée`QldbDriver` et

une`PooledQldbDriver`. Comme son nom l'indique, il`PooledQldbDriver` gère un pool de sessions qui sont réutilisées dans toutes les transactions.

Sur la base des commentaires des utilisateurs, les développeurs préfèrent utiliser la fonctionnalité de mise en commun et ses avantages par défaut, plutôt que d'utiliser le pilote standard. Nous avons donc supprimé`PooledQldbDriver` et déplacé la fonctionnalité de regroupement de sessions vers`QldbDriver`. Cette modification est incluse dans la dernière version de chaque pilote (par exemple, Java v2.0.0).

Le pilote propose trois niveaux d'abstractions :

- Pilote (implémentation de pilotes groupés) : abstraction de haut niveau. Le chauffeur gère et gère un pool de sessions. Lorsque vous demandez au chauffeur d'exécuter une transaction, il choisit une session dans le pool et l'utilise pour exécuter la transaction. Si la transaction échoue en raison d'une erreur de session (`InvalidSessionException`), le pilote utilise une autre session pour réessayer la transaction. Essentiellement, le conducteur offre une expérience de session entièrement gérée.
- Session : un niveau en dessous de l'abstraction du conducteur. Une session est contenue dans un pool et le pilote gère le cycle de vie de la session. En cas d'échec d'une transaction, le conducteur réessaie jusqu'à un certain nombre de tentatives à l'aide de la même session. Mais si la session rencontre une erreur (`InvalidSessionException`), QLDB la supprime en interne. Le chauffeur est ensuite chargé d'obtenir une autre session du pool pour réessayer la transaction.
- Transaction : niveau d'abstraction le plus bas. Une transaction est contenue dans une session, et la session gère le cycle de vie de la transaction. La session est chargée de réessayer la transaction en cas d'erreur. La session garantit également qu'elle ne divulgue pas une transaction ouverte qui n'a pas été validée ou annulée.

Dans la dernière version de chaque pilote, vous pouvez effectuer des opérations au niveau d'abstraction du pilote uniquement. Vous n'avez aucun contrôle direct sur les sessions et les transactions individuelles (c'est-à-dire qu'aucune opération d'API ne permet de démarrer manuellement une nouvelle session ou transaction).

Regroupement de sessions et logique transactionnelle

La dernière version de chaque pilote ne fournit plus d'implémentation non groupée de l'objet pilote. Par défaut, l'`QldbDriver`objet gère le pool de sessions. Lorsque vous passez un appel pour exécuter une transaction, le chauffeur effectue les étapes suivantes :

1. Le conducteur vérifie si la limite du pool de sessions a été atteinte. Si tel est le cas, le conducteur émet immédiatement une`NoSessionAvailable` exception. Autrement, il passe à l'étape suivante.
2. Le pilote vérifie si une session est disponible dans le pool.
   - Si une session est disponible dans le pool, le pilote l'utilise pour exécuter une transaction.
   - Si aucune session n'est disponible dans le pool, le pilote crée une nouvelle session et l'utilise pour exécuter une transaction.
3. Une fois que le pilote a ouvert une session à l'étape 2, il appelle l'`execute`opération sur l'instance de session.
4. Pendant le`execute` fonctionnement de la session, le chauffeur essaie de démarrer une transaction en appelant`startTransaction`.
   - Si la session n'est pas valide, l'`startTransaction`appel échoue et le pilote revient à l'étape 1.
   - Si l'`startTransaction`appel aboutit, le chauffeur passe à l'étape suivante.
5. Le pilote exécute votre expression lambda. Cette expression lambda peut contenir un ou plusieurs appels pour exécuter des instructions partiQL. Une fois que l'expression lambda a fini de s'exécuter sans erreur, le pilote procède à la validation de la transaction.
6. La validation de la transaction peut avoir l'un des deux résultats suivants :
   - La validation est réussie et le pilote reprend le contrôle du code de votre application.
   - La validation échoue en raison d'un conflit de contrôle de la concurrence (OCC) optimiste. Dans ce cas, le conducteur réessaie les étapes 4 à 6 en utilisant la même session. Le nombre maximal de nouvelles tentatives est paramétrable dans le code de votre application. La limite par défaut est4.

> **Note**
>
> Si un`InvalidSessionException` est lancé au cours des étapes 4 à 6, le chauffeur marque la session comme fermée et revient à l'étape 1 pour réessayer la transaction.
> Si une autre exception est déclenchée au cours des étapes 4 à 6, le conducteur vérifie si l'exception peut être réessayée. Si tel est le cas, il réessaie la transaction jusqu'au nombre de tentatives spécifié. Si ce n'est pas le cas, il propage (fait des bulles et génère) l'exception dans le code de votre application.

Retour des sessions à la piscine

Si un`InvalidSessionException` se produit à tout moment au cours d'une transaction active, le conducteur ne renvoie pas la session dans le pool. QLDB supprime la session à la place et le pilote obtient une autre session du pool. Dans tous les autres cas, le pilote renvoie la session dans le pool.

Recommandations de pilotes Amazon QLDB

Cette section décrit les meilleures pratiques de configuration et d'utilisation du pilote Amazon QLDB pour n'importe quelle langue prise en charge. Les exemples de code fournis concernent spécifiquement Java.

Ces recommandations s'appliquent à la plupart des cas d'utilisation courants, mais une taille unique ne convient pas à tous. Utilisez les recommandations suivantes comme bon vous semble approprié pour votre application.

Rubriques

Configuration de QldbDriver objet

Le`QldbDriver`gère les connexions à votre livre en maintenant un pool desessionsqui sont réutilisés dans toutes les transactions. UNsessionreprésente une connexion unique au grand livre. QLDB prend en charge une transaction active par session.

> **Important**
>
> Pour les anciennes versions de pilotes, la fonctionnalité de pool de sessions se trouve toujours dans le`PooledQldbDriver`objet et non de`QldbDriver`. Si vous utilisez l'une des versions suivantes, remplacez toutes les mentions de`QldbDriver`avec`PooledQldbDriver`pour le reste de ce sujet.

Pilote	Version
Java	`1.1.0`ou antérieures
.NET	`0.1.0-beta`
Node.js	`1.0.0-rc.1` ou antérieures
Python	`2.0.2`ou antérieures

Le`PooledQldbDriver`l'objet est obsolète dans la dernière version des pilotes. Nous vous recommandons de procéder à une mise à niveau vers la dernière version et de convertir toutes les instances de`PooledQldbDriver`pour`QldbDriver`.

Configuration QldbDriver en tant qu'objet global

Pour optimiser l'utilisation des pilotes et des sessions, assurez-vous qu'une seule instance globale du pilote existe dans votre instance d'application. Par exemple, en Java, vous pouvez utiliserInjection de dépendanceframeworks tels quePrintemps,Google Guide, ouDague. L'exemple de code suivant montre comment configurer.`QldbDriver`en singleton.

```
@Singleton
public QldbDriver qldbDriver (AWSCredentialsProvider credentialsProvider,
                                  @Named(LEDGER_NAME_CONFIG_PARAM) String ledgerName)
 {
    QldbSessionClientBuilder builder = QldbSessionClient.builder();
    if (null != credentialsProvider) {
        builder.credentialsProvider(credentialsProvider);
    }
    return QldbDriver.builder()
            .ledger(ledgerName)
            .transactionRetryPolicy(RetryPolicy
                .builder()
                .maxRetries(3)
                .build())
            .sessionClientBuilder(builder)
            .build();
}
```

Configurer les tentatives de nouvelle tentative

Le pilote retente automatiquement les transactions lorsque des exceptions transitoires courantes (telles que`SocketTimeoutException`ou`NoHttpResponseException`) se produisent. Pour définir le nombre maximal de nouvelles tentatives, vous pouvez utiliser l'option`maxRetries`paramètre de la`transactionRetryPolicy`objet de configuration lors de la création d'une instance de`QldbDriver`. (Pour les anciennes versions de pilotes répertoriées dans la section précédente, utilisez le`retryLimit`paramètre de`PooledQldbDriver`.)

La valeur par défaut de `maxRetries` est `4`.

Erreurs côté du client, telles que`InvalidParameterException`ne peut pas être réessayé. Lorsqu'elles se produisent, la transaction est interrompue, la session est renvoyée au pool et l'exception est renvoyée au client du pilote.

Configuration du nombre maximal de sessions et de transactions simultanées

Le nombre maximal de sessions de livres utilisées par une instance de`QldbDriver`pour exécuter des transactions est défini par son`maxConcurrentTransactions` Paramètre . (Pour les anciennes versions de pilotes répertoriées dans la section précédente, cela est défini par le`poolLimit`paramètre de`PooledQldbDriver`.)

Cette limite doit être supérieure à zéro et inférieure ou égale au nombre maximal de connexions HTTP ouvertes que le client de session autorise, tel que défini par le paramètre spécifique.AWSSDK. Par exemple, en Java, le nombre maximal de connexions est défini dans leClientConfigurationobjet.

La valeur par défaut de`maxConcurrentTransactions`est le paramètre de connexion maximal de votreAWSSDK.

Lorsque vous configurez le`QldbDriver`dans votre application, prenez en compte les considérations de dimensionnement suivantes :

- Votre pool doit toujours avoir au moins autant de sessions que le nombre de transactions exécutées simultanément que vous prévoyez d'avoir.
- Dans un modèle multithread où un thread de superviseur délègue à des threads de travail, le pilote doit avoir au moins autant de sessions que le nombre de threads de travail. Sinon, à la charge maximale, les threads seront en attente d'une session disponible.
- La limite de service des sessions actives simultanées par livre est définie dansQuotas et limites d'Amazon QLDB. Assurez-vous que vous ne configurez pas plus que cette limite de sessions simultanées pour être utilisées pour un seul livre sur tous les clients.

Nouvelles tentatives en cas d'exception

Lorsque vous tentez de nouveau les exceptions qui se produisent dans QLDB, tenez compte des recommandations suivantes.

Nouvelles tentatives d'exception OccConflictException

Contrôle optimiste de la concurrenceLes exceptions de conflit (OCC) se produisent lorsque les données auxquelles la transaction accède ont changé depuis le début de la transaction. QLDB lance cette exception en essayant de valider la transaction. Le conducteur retente la transaction jusqu'à autant de fois que`maxRetries`est configuré.

Pour plus d'informations sur OCC et sur les meilleures pratiques d'utilisation des index pour limiter les conflits OCC, voirModèle de mise QLDB simultansimultansimultansimultansimultansimultansimultansimultansimultanéité.

Réessayer d'autres exceptions en dehors de QLDBDriver

Pour réessayer une transaction en dehors du pilote lorsque des exceptions personnalisées définies par l'application sont lancées pendant l'exécution, vous devez encapsuler la transaction. Par exemple, en Java, le code suivant montre comment utiliser l'optionRélience 4Jpour réessayer une transaction dans QLDB.

```
private final RetryConfig retryConfig = RetryConfig.custom()
        .maxAttempts(MAX_RETRIES)
        .intervalFunction(IntervalFunction.ofExponentialRandomBackoff())
        // Retry this exception
        .retryExceptions(InvalidSessionException.class, MyRetryableException.class)
        // But fail for any other type of exception extended from RuntimeException
        .ignoreExceptions(RuntimeException.class)
        .build();

// Method callable by a client
public void myTransactionWithRetries(Params params) {
    Retry retry = Retry.of("registerDriver", retryConfig);

    Function<Params, Void> transactionFunction = Retry.decorateFunction(
            retry,
            parameters ->  transactionNoReturn(params));
    transactionFunction.apply(params);
}
```

```
private Void transactionNoReturn(Params params) {
    try (driver.execute(txn -> {
            // Transaction code
        });
    }
    return null;
}
```

> **Note**
>
> La nouvelle tentative d'une transaction en dehors du pilote QLDB a un effet multiplicateur. Par exemple, si`QldbDriver`est configuré pour réessayer trois fois, et la logique de nouvelle tentative personnalisée à nouveau trois fois, la même transaction peut être retentée jusqu'à neuf fois.

Indempotence des transactions

Comme meilleure pratique, rendez vos transactions d'écriture idépotentes afin d'éviter tout effet secondaire inattendu en cas de nouvelles tentatives. Une transaction estidempotentss'il peut s'exécuter plusieurs fois et produire des résultats identiques à chaque fois.

Pour en savoir plus, veuillez consulter la section Modèle de mise QLDB simultansimultansimultansimultansimultansimultansimultansimultansimultanéité.

Optimisation des performances

Pour optimiser les performances lorsque vous exécutez des transactions à l'aide du pilote, tenez compte des considérations suivantes :

- Le`execute`l'opération fait toujours un minimum de trois`SendCommand`Appels API vers QLDB, y compris les commandes suivantes :
  1. `StartTransaction`
  2. `ExecuteStatement`

     Cette commande est appelée pour chaque instruction PartiQL que vous exécutez dans le`execute`block.
  3. `CommitTransaction`

Tenez compte du nombre total d'appels d'API effectués lorsque vous calculez la charge de travail globale de votre application.

- En général, nous recommandons de commencer par un graveur monothread et d'optimiser les transactions en regroupant plusieurs instructions au sein d'une seule transaction. Maximisez les quotas sur la taille des transactions, la taille du document et le nombre de documents par transaction, comme défini dansQuotas et limites d'Amazon QLDB.
- Si le traitement par lots n'est pas suffisant pour les charges de transactions volumineuses, vous pouvez essayer le multithreading en ajoutant des rédacteurs supplémentaires. Toutefois, vous devez examiner attentivement les exigences de votre application pour le séquençage des documents et des transactions, ainsi que la complexité supplémentaire que cela introduit.

Exécution de plusieurs instructions par transaction

Comme décrit dans lasection précédente, vous pouvez exécuter plusieurs relevés par transaction pour optimiser les performances de votre application. Dans l'exemple de code suivant, vous interrogez une table, puis mettez à jour un document de cette table dans une transaction. Pour ce faire, vous transmettez une expression lambda à la`execute`.

Java

```
// This code snippet is intentionally trivial. In reality you wouldn't do this
 because you'd
// set your UPDATE to filter on vin and insured, and check if you updated something
 or not.
public static boolean InsureCar(QldbDriver qldbDriver, final String vin) {
    final IonSystem ionSystem = IonSystemBuilder.standard().build();
    final IonString ionVin = ionSystem.newString(vin);

    return qldbDriver.execute(txn -> {
        Result result = txn.execute(
                "SELECT insured FROM Vehicles WHERE vin = ? AND insured = FALSE",
                ionVin);
        if (!result.isEmpty()) {
            txn.execute("UPDATE Vehicles SET insured = TRUE WHERE vin = ?", ionVin);
            return true;
        }
        return false;
    });
```

```
}
```

.NET

```
// This code snippet is intentionally trivial. In reality you wouldn't do this
 because you'd
// set your UPDATE to filter on vin and insured, and check if you updated something
 or not.
public static async Task<bool> InsureVehicle(IAsyncQldbDriver driver, string vin)
{
    ValueFactory valueFactory = new ValueFactory();
    IIonValue ionVin = valueFactory.NewString(vin);

    return await driver.Execute(async txn =>
    {
        // Check if the vehicle is insured.
        Amazon.QLDB.Driver.IAsyncResult result = await txn.Execute(
            "SELECT insured FROM Vehicles WHERE vin = ? AND insured = FALSE",
 ionVin);

        if (await result.CountAsync() > 0)
        {
            // If the vehicle is not insured, insure it.
            await txn.Execute(
                "UPDATE Vehicles SET insured = TRUE WHERE vin = ?", ionVin);
            return true;
        }
        return false;
    });
}
```

Go

```
// This code snippet is intentionally trivial. In reality you wouldn't do this
 because you'd
// set your UPDATE to filter on vin and insured, and check if you updated something
 or not.
func InsureCar(driver *qldbdriver.QLDBDriver, vin string) (bool, error) {
    insured, err := driver.Execute(context.Background(), func(txn
 qldbdriver.Transaction) (interface{}, error) {

        result, err := txn.Execute(
            "SELECT insured FROM Vehicles WHERE vin = ? AND insured = FALSE", vin)
```

```
        if err != nil {
            return false, err
        }

        hasNext := result.Next(txn)
        if !hasNext && result.Err() != nil {
            return false, result.Err()
        }

        if hasNext {
            _, err = txn.Execute(
                "UPDATE Vehicles SET insured = TRUE WHERE vin = ?", vin)
            if err != nil {
                return false, err
            }
            return true, nil
        }
        return false, nil
    })
    if err != nil {
        panic(err)
    }

    return insured.(bool), err
}
```

Node.js

```
// This code snippet is intentionally trivial. In reality you wouldn't do this
 because you'd
// set your UPDATE to filter on vin and insured, and check if you updated something
 or not.
async function insureCar(driver: QldbDriver, vin: string): Promise<boolean> {

    return await driver.executeLambda(async (txn: TransactionExecutor) => {
        const results: dom.Value[] = (await txn.execute(
            "SELECT insured FROM Vehicles WHERE vin = ? AND insured = FALSE",
 vin)).getResultList();

        if (results.length > 0) {
            await txn.execute(
                "UPDATE Vehicles SET insured = TRUE WHERE vin = ?", vin);
            return true;
```

```
        }
        return false;
    });
};
```

Python

```
# This code snippet is intentionally trivial. In reality you wouldn't do this
 because you'd
# set your UPDATE to filter on vin and insured, and check if you updated something
 or not.

def do_insure_car(transaction_executor, vin):
    cursor = transaction_executor.execute_statement(
        "SELECT insured FROM Vehicles WHERE vin = ? AND insured = FALSE", vin)
    first_record = next(cursor, None)
    if first_record:
        transaction_executor.execute_statement(
            "UPDATE Vehicles SET insured = TRUE WHERE vin = ?", vin)
        return True
    else:
        return False

def insure_car(qldb_driver, vin_to_insure):
    return qldb_driver.execute_lambda(
        lambda executor: do_insure_car(executor, vin_to_insure))
```

Le chauffeur`execute`l'opération démarre implicitement une session et une transaction dans cette session. Chaque instruction que vous exécutez dans l'expression lambda est encapsulée dans la transaction. Une fois toutes les instructions exécutées, le pilote valide automatiquement la transaction. Si une instruction échoue après que la limite de nouvelle tentative automatique est épuisée, la transaction est interrompue.

Propager les exceptions dans une transaction

Lorsque vous exécutez plusieurs instructions par transaction, nous ne recommandons généralement pas d'catch et d'avaler des exceptions au sein de la transaction.

Par exemple, en Java, le programme suivant capture n'importe quelle instance de`RuntimeException`, enregistre l'erreur et continue. Cet exemple de code est considéré comme

une mauvaise pratique car la transaction réussit même lorsque leUPDATEl'instruction échoue. Par conséquent, le client peut supposer que la mise à jour a réussi alors que ce n'est pas le cas.

> ⚠ Warning
>
> N'utilisez pas cet exemple de code. Il est fourni pour montrer un exemple anti-pattern considéré comme une mauvaise pratique.

```
// DO NOT USE this code example because it is considered bad practice
public static void main(final String... args) {
    ConnectToLedger.getDriver().execute(txn -> {
        final Result selectTableResult = txn.execute("SELECT * FROM Vehicle WHERE VIN
 ='123456789'");
        // Catching an error inside the transaction is an anti-pattern because the
 operation might
        // not succeed.
        // In this example, the transaction succeeds even when the update statement
 fails.
        // So, the client might assume that the update succeeded when it didn't.
        try {
            processResults(selectTableResult);
            String model = // some code that extracts the model
            final Result updateResult = txn.execute("UPDATE Vehicle SET model = ? WHERE
 VIN = '123456789'",
                    Constants.MAPPER.writeValueAsIonValue(model));
        } catch (RuntimeException e) {
            log.error("Exception when updating the Vehicle table {}", e.getMessage());
        }
    });
    log.info("Vehicle table updated successfully.");
}
```

Propagez plutôt l'exception (bulle vers le haut). Si une partie de la transaction échoue, laissez le`execute`l'opération interrompt la transaction afin que le client puisse gérer l'exception en conséquence.

Comprendre la stratégie de nouvelle tentative avec le pilote dans Amazon QLDB

Le pilote Amazon QLDB utilise une stratégie de nouvelle tentative pour gérer les exceptions transitoires en retentant de manière transparente une transaction ayant échoué. Ces exceptions, telles que`CapacityExceededException`et`RateExceededException`, se corrigent généralement après une période donnée. Si la transaction qui a échoué à l'exception fait l'objet d'une nouvelle tentative après un retard approprié, elle est susceptible de réussir. Cela contribue à améliorer la stabilité de l'application utilisant QLDB.

Rubriques

Types d'erreurs réessayables

Le pilote retente automatiquement une transaction si et uniquement si l'une des exceptions suivantes se produit au cours d'une opération au sein de cette transaction :

- Exception de capacité dépassée— Retourné lorsque la demande dépasse la capacité de traitement du livre.
- Exception de session non valide— Retourné lorsqu'une session n'est plus valide ou si la session n'existe pas.
- LimitExceededException— Retourné si une limite de ressources telle que le nombre de sessions actives est dépassée.
- Exception de conflit OCC— Retourné lorsqu'une transaction ne peut pas être écrite dans le journal en raison d'un échec de la phase de vérification decontrôle optimiste de la concurrence(OCC).
- Dépassement du taux d'exception— Retourné lorsque le taux de requêtes dépasse le débit autorisé.

Stratégie de nouvelle tentative d'essai par défaut

La stratégie de nouvelle tentative consiste en une condition de nouvelle tentative et une stratégie d'annulation. La condition de nouvelle tentative définit quand une transaction doit être réessayée, tandis que la stratégie de backoff définit la durée d'attente avant de réessayer la transaction.

Lors de la création d'une instance du pilote, la stratégie de nouvelle tentative par défaut spécifie de réessayer jusqu'à quatre fois et d'utiliser une stratégie de backoff exponentielle. La stratégie de retour exponentiel utilise un délai minimum de 10 millisecondes et un délai maximal de 5000 millisecondes, avec une gigue égale. Si la transaction ne peut pas être validée avec succès dans la stratégie de nouvelle tentative, nous vous recommandons d'essayer la transaction à un autre moment.

Le concept consiste à utiliser des temps d'attente progressivement plus longs entre les tentatives en cas de réponses d'erreur consécutives. Pour plus d'informations, consultez le .AWSbillet de blogInstabilité et backoff exponentiel.

Erreurs courantes liées au pilote Amazon QLDB

Cette section décrit les erreurs d'exécution qui peuvent être générées par le pilote Amazon QLDB lors de l'interaction avec l'API de session QLDB.

Voici la liste des exceptions courantes qui sont renvoyées par le pilote. Chaque exception inclut le message d'erreur spécifique, suivi d'une brève description et de suggestions de solutions possibles.

CapacityExceededException

Message : Capacité dépassée

Amazon QLDB a rejeté la demande car elle dépassait la capacité de traitement du registre. QLDB impose une limite de dimensionnement interne par registre afin de maintenir l'intégrité et les performances du service. Cette limite varie en fonction de la charge de travail de chaque demande individuelle. Par exemple, une demande peut avoir une charge de travail accrue si elle effectue des transactions de données inefficaces, telles que des analyses de tables résultant d'une requête qualifiée non indexée.

Nous vous recommandons d'attendre avant de réessayer la demande. Si votre application rencontre régulièrement cette exception, optimisez vos relevés et diminuez le taux et le volume des demandes que vous envoyez au registre. Parmi les exemples d'optimisation des instructions, citons l'exécution d'un nombre réduit d'instructions par transaction et le réglage des index de vos tables. Pour savoir comment optimiser les instructions et éviter de scanner des tableaux, consultezOptimisation des performances des données.

Nous vous recommandons également d'utiliser la dernière version du pilote QLDB. Le pilote applique une politique de nouvelle tentative par défaut qui utilise Exponential Backoff and Jitter pour réessayer automatiquement en cas d'exceptions telles que celle-ci. Le concept de retard

exponentiel consiste à utiliser des temps d'attente progressivement plus longs entre les nouvelles tentatives pour des réponses d'erreur consécutives.

InvalidSessionException

Message : L'*`ID de transaction`* a expiré

Une transaction a dépassé sa durée de vie maximale. Une transaction peut s'exécuter pendant 30 secondes au maximum avant d'être validée. Passé ce délai, tout travail effectué sur la transaction est rejeté et QLDB ferme la session. Cette limite protège le client contre les fuites de sessions en démarrant des transactions sans les valider ni les annuler.

S'il s'agit d'une exception courante dans votre application, il est probable que les transactions prennent tout simplement trop de temps à s'exécuter. Si l'exécution des transactions prend plus de 30 secondes, optimisez vos relevés pour accélérer les transactions. Parmi les exemples d'optimisation des instructions, citons l'exécution d'un nombre réduit d'instructions par transaction et le réglage des index de vos tables. Pour plus d'informations, veuillez consulter Optimisation des performances des données.

InvalidSessionException

Message : L'*`sessionId`* a expiré

QLDB a supprimé la session car elle a dépassé sa durée de vie totale maximale. QLDB supprime les sessions au bout de 13 à 17 minutes, qu'il s'agisse d'une transaction active. Les sessions peuvent être perdues ou altérées pour diverses raisons, telles qu'une panne matérielle, une panne réseau ou le redémarrage d'applications. QLDB impose donc une durée de vie maximale aux sessions afin de garantir la résilience du logiciel client en cas d'échec de session.

Si vous rencontrez cette exception, nous vous recommandons d'ouvrir une nouvelle session et de réessayer la transaction. Nous vous recommandons également d'utiliser la dernière version du pilote QLDB, qui gère le pool de sessions et son intégrité pour le compte de l'application.

InvalidSessionException

Message : Aucune session de ce type

Le client a essayé d'effectuer des transactions avec QLDB en utilisant une session qui n'existe pas. En supposant que le client utilise une session qui existait auparavant, il se peut que la session n'existe plus pour l'une des raisons suivantes :

- Si une session est impliquée dans une défaillance interne du serveur (c'est-à-dire une erreur avec le code de réponse HTTP 500), QLDB peut choisir de supprimer complètement la session

plutôt que d'autoriser le client à effectuer des transactions avec une session dont l'état est incertain. Ensuite, toute nouvelle tentative sur cette session échoue avec cette erreur.

- Les sessions expirées sont finalement oubliées par QLDB. Ensuite, toute tentative de poursuite de l'utilisation de la session entraîne cette erreur, plutôt que la première`InvalidSessionException`.

Si vous rencontrez cette exception, nous vous recommandons d'ouvrir une nouvelle session et de réessayer la transaction. Nous vous recommandons également d'utiliser la dernière version du pilote QLDB, qui gère le pool de sessions et son intégrité pour le compte de l'application.

RateExceededException

Message : Le taux a été dépassé

QLDB a limité un client en fonction de l'identité de l'appelant. QLDB applique la limitation par région et par compte à l'aide d'un algorithme de limitation des compartiments à jetons. QLDB le fait pour améliorer les performances du service et garantir une utilisation équitable pour tous les clients. Par exemple, la tentative d'acquisition d'un grand nombre de sessions simultanées à l'aide de l'`StartSessionRequest`opération peut entraîner une limitation.

Pour préserver l'intégrité de votre application et atténuer toute nouvelle limitation, vous pouvez réessayer cette exception en utilisant Exponential Backoff and Jitter. Le concept de retard exponentiel consiste à utiliser des temps d'attente progressivement plus longs entre les nouvelles tentatives pour des réponses d'erreur consécutives. Nous vous recommandons d'utiliser la dernière version du pilote QLDB. Le pilote applique une politique de nouvelle tentative par défaut qui utilise une interruption et une instabilité exponentielles pour réessayer automatiquement en cas d'exceptions telles que celle-ci.

La dernière version du pilote QLDB peut également être utile si votre application est constamment bloquée par QLDB pour les`StartSessionRequest` appels. Le pilote gère un pool de sessions qui sont réutilisées pour toutes les transactions, ce qui peut contribuer à réduire le nombre d'`StartSessionRequest`appels effectués par votre application. Pour demander une augmentation des limitations des API, veuillez contacter le AWS SupportCentre.

LimitExceededException

Message : La limite de session a été dépassée

Un registre a dépassé son quota (également appelé limite) en ce qui concerne le nombre de sessions actives. Ce quota est défini dansQuotas et limites d'Amazon QLDB. Le nombre de

sessions actives d'un registre finit par être constant, et les registres qui se situent régulièrement à proximité du quota peuvent régulièrement connaître cette exception.

Pour préserver l'intégrité de votre application, nous vous recommandons de réessayer cette exception. Pour éviter cette exception, assurez-vous de ne pas avoir configuré plus de 1 500 sessions simultanées à utiliser pour un seul registre pour tous les clients. Par exemple, vous pouvez utiliser la maxConcurrentTransactionsméthode du pilote Amazon QLDB pour Java afin de configurer le nombre maximum de sessions disponibles dans une instance de pilote.

QldbClientException

Message : Un résultat diffusé en continu n'est valide que lorsque la transaction parent est ouverte

La transaction est fermée et ne peut pas être utilisée pour récupérer les résultats depuis QLDB. Une transaction est clôturée lorsqu'elle est validée ou annulée.

Cette exception se produit lorsque le client travaille directement avec l'`Transaction`objet et essaie de récupérer les résultats de QLDB après avoir validé ou annulé une transaction. Pour atténuer ce problème, le client doit lire les données avant de clôturer la transaction.

Démarrer avec Amazon QLDB à l'aide d'un exemple de didacticiel d'application

Dans ce didacticiel, vous utilisez le pilote Amazon QLDB avec unAWS SDK pour créer un registre QLDB et le renseigner avec des exemples de données. Le pilote permet à votre application d'interagir avec QLDB à l'aide de l'API de données transactionnelles. LeAWS SDK prend en charge l'interaction avec l'API de gestion des ressources QLDB.

L'exemple de registre que vous créez dans ce scénario est une base de données du département des véhicules automobiles (DMV) qui contient les informations historiques complètes sur les immatriculations de véhicules. Les rubriques suivantes expliquent comment ajouter des immatriculations de véhicules, les modifier et consulter l'historique des modifications apportées à ces immatriculations. Ce guide explique également comment vérifier cryptographiquement un document d'enregistrement. Il se termine par le nettoyage des ressources et la suppression du registre d'exemple.

Cet exemple de didacticiel d'application est disponible pour les langages de programmation suivants.

Rubriques

- Tutoriel Amazon QLDB
- Tutoriel Amazon QLDB Node.js
- Tutoriel Amazon QLDB Python

Tutoriel Amazon QLDB

Dans cette implémentation de l'exemple d'application du didacticiel, vous utilisez le pilote Amazon QLDB avec leAWS SDK for Java pour créer un registre QLDB et le remplir avec des exemples de données.

En lien avec ce didacticiel, vous pouvez vous référer à la RéférenceAWS SDK for Java d'API pour les opérations d'API Pour les opérations sur les données transactionnelles, vous pouvez vous référer au pilote QLDB pour Java API Reference.

> **Note**
>
> Le cas échéant, certaines étapes du didacticiel comportent des commandes ou des exemples de code différents pour chaque version majeure prise en charge du pilote QLDB pour Java.

Rubriques

- Installation de l'exemple d'application Java Amazon QLDB
- Étape 1 : Créer un registre
- Étape 2 : Tester de la connectivité au registre
- Étape 3 : Création de tables, d'index et d'exemples de données
- Étape 4 : interroger les tables d'un registre
- Étape 5 : Modifier des documents dans un registre
- Étape 6 : Afficher l'historique des révisions d'un document
- Étape 7 : Vérification d'un document dans un registre
- Étape 8 : Exporter et valider les données du journal dans un registre
- Étape 9 (facultatif) : Nettoyer les ressources

Installation de l'exemple d'application Java Amazon QLDB

Cette section explique comment installer et exécuter l'exemple d'application Amazon QLDB fourni pour le didacticiel step-by-step Java. Le cas d'utilisation de cet exemple d'application est une base de données du ministère des véhicules automobiles (DMV) qui permet de suivre l'historique complet des immatriculations de véhicules.

L'exemple d'application DMV pour Java est open source dans le GitHub référentiel aws-samples/amazon-qldb-dmv-sample -java.

Prérequis

Avant de commencer, vérifiez que vous exécutez le pilote QLDB pour JavaPrérequis. Cela inclut les éléments suivants :

1. S'inscrire àAWS.
2. Créez un utilisateur avec les autorisations QLDB appropriées. Pour effectuer toutes les étapes de ce didacticiel, vous devez disposer d'un accès administratif complet à votre ressource de registre via l'API QLDB.
3. Si vous utilisez un IDE autre queAWS Cloud9, installez Java et accordez un accès programmatique pour le développement.

Installation

Les étapes suivantes décrivent comment télécharger et configurer l'exemple d'application dans un environnement de développement local. Vous pouvez également automatiser la configuration de l'exemple d'application en utilisantAWS Cloud9 comme IDE unAWS CloudFormation modèle pour provisionner vos ressources de développement.

Environnement de développement local

Ces instructions décrivent comment télécharger et installer l'exemple d'application Java QLDB à l'aide de vos propres ressources et de votre environnement de développement.

Pour télécharger et exécuter l'exemple d'application

1. Saisissez la commande suivante pour cloner l'exemple d'application GitHub.

2.x

```
git clone https://github.com/aws-samples/amazon-qldb-dmv-sample-java.git
```

1.x

```
git clone -b v1.2.0 https://github.com/aws-samples/amazon-qldb-dmv-sample-
java.git
```

Ce package inclut la configuration de Gradle et le code complet duTutoriel Java.

2. Chargez et exécutez l'application fournie.
   - Si vous utilisez Eclipse :
     a. Démarrez Eclipse et, dans le menu Eclipse, choisissez Fichier, Importer, puis Projet Gradle existant.
     b. Dans le répertoire racine du projet, parcourez et sélectionnez le répertoire de l'application qui contient le`build.gradle` fichier. Choisissez ensuite Terminer pour utiliser les paramètres Gradle par défaut pour l'importation.
     c. Vous pouvez essayer d'exécuter le`ListLedgers` programme à titre d'exemple. Ouvrez le menu contextuel (clic droit) du`ListLedgers.java` fichier et choisissez Exécuter en tant qu'application Java.
   - Si vous utilisez IntelliJ :
     a. Démarrez IntelliJ et, dans le menu IntelliJ, choisissez Fichier, puis Ouvrir.
     b. Dans le répertoire racine du projet, parcourez et sélectionnez le répertoire de l'application qui contient le`build.gradle` fichier. Ensuite, choisissez OK. Vous pouvez conserver les paramètres par défaut, puis sélectionner à nouveau OK.
     c. Vous pouvez essayer d'exécuter le`ListLedgers` programme à titre d'exemple. Ouvrez le menu contextuel (clic droit) du`ListLedgers.java` fichier et choisissez Exécuter «ListLedgers ».
3. PassezÉtape 1 : Créer un registre à démarrer le didacticiel et à créer un registre.

AWS Cloud9

Ces instructions décrivent comment automatiser la configuration de l'exemple d'application d'enregistrement de véhicules Amazon QLDB pour Java, en l'utilisant AWS Cloud9comme IDE. Dans ce guide, vous utilisez un AWS CloudFormationmodèle pour fournir vos ressources de développement.

Pour plus d'informations sur AWS Cloud9, consultez le AWS Cloud9Guide de l'utilisateur . Pour en savoir plus sur AWS CloudFormation, veuillez consulter le Guide de l'utilisateur AWS CloudFormation.

Rubriques

- Partie 1 : Approvisionnez vos ressources
- Partie 2 : Configurer votre IDE
- Partie 3 : Exécuter l'exemple d'application QLDB DMV

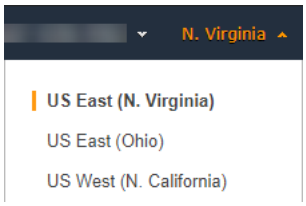
Partie 1 : Approvisionnez vos ressources

Au cours de cette première étape, vousAWS CloudFormation devez fournir les ressources requises pour configurer votre environnement de développement à l'aide de l'exemple d'application Amazon QLDB.

Pour ouvrir laAWS CloudFormation console et charger l'exemple de modèle d'application QLDB

1. Connectez-vous à la AWS Management Console et ouvrez la console AWS CloudFormation à l'adresse https://console.aws.amazon.com/cloudformation.

   Basculez vers une région qui prend en charge QLDB. Pour obtenir la liste complète, consultez les points de terminaison et quotas Amazon QLDB dans le Références générales AWS. La capture d'écran suivanteAWS Management Console montre US East (Virginie du Nord) comme sélectionnéRégion AWS.

2. Sur laAWS CloudFormation console, choisissez Créer une pile, puis choisissez Avec de nouvelles ressources (standard).

3. Sur la page Créer une pile, sous Spécifier le modèle, choisissez l'URL Amazon S3.
4. Entrez l'URL suivante, puis choisissez Next.

```
https://amazon-qldb-assets.s3.amazonaws.com/templates/QLDB-DMV-SampleApp.yml
```

5. Entrez un nom de pile (tel que**`qldb-sample-app`**), puis choisissez Next.
6. Vous pouvez ajouter les balises appropriées et conserver les options par défaut. Sélectionnez ensuite Next (Suivant).
7. Vérifiez les paramètres de votre pile, puis choisissez Créer une pile. La fin duAWS CloudFormation script peut prendre quelques minutes.

   Ce script fournit votreAWS Cloud9 environnement à une instance Amazon Elastic Compute Cloud (Amazon EC2) associée que vous utilisez pour exécuter l'exemple d'application QLDB de ce didacticiel. Il clone également le référentiel aws-samples/amazon-qldb-dmv-sample -java depuis GitHub votre environnement deAWS Cloud9 développement.

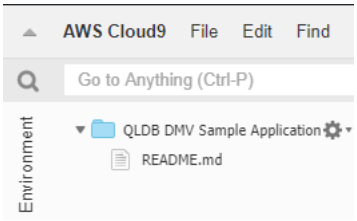
Partie 2 : Configurer votre IDE

Dans cette étape, vous terminez la configuration de votre environnement de développement cloud. Vous téléchargez et exécutez un script shell fourni pour configurer votreAWS Cloud9 IDE avec les dépendances des exemples d'application.

Pour configurer votreAWS Cloud9 environnement

1. Ouvrez laAWS Cloud9 console à l'adresse https://console.aws.amazon.com/cloud9/.
2. Sous Vos environnements, recherchez la carte correspondant à l'environnement nommée QLDB DMV Sample Application, puis choisissez Open IDE. Le chargement de votre environnement peut prendre une minute lors du lancement de l'instance EC2 sous-jacente.

   VotreAWS Cloud9 environnement est préconfiguré avec les dépendances du système dont vous avez besoin pour exécuter le didacticiel. Dans le volet de navigation Environment de votre console, vérifiez qu'un dossier nommé s'afficheQLDB DMV Sample Application. La capture d'écran suivante de laAWS Cloud9 console montre le volet de dossiers d'environnement QLDB DMV Sample Application.



Si aucun volet de navigation ne s'affiche, basculez sur l'onglet Environnement sur le côté gauche de votre console. Si aucun dossier ne s'affiche dans le volet, activez Afficher la racine de l'environnement à l'aide de l'icône des paramètres ( ).

3. Dans le volet inférieur de votre console, vous devriez voir une fenêtre de`bash` terminal ouverte. Si ce n'est pas le cas, choisissez Nouveau terminal dans le menu Fenêtre en haut de la console.
4. Ensuite, téléchargez et exécutez un script de configuration pour installer OpenJDK 8 et, le cas échéant, consultez la branche appropriée dans le référentiel Git. Dans leAWS Cloud9 terminal que vous avez créé lors de l'étape précédente, exécutez les deux commandes suivantes dans l'ordre :

   2.x

   ```
   aws s3 cp s3://amazon-qldb-assets/setup-scripts/dmv-setup-v2.sh .
   ```

   ```
   sh dmv-setup-v2.sh
   ```

   1.x

   ```
   aws s3 cp s3://amazon-qldb-assets/setup-scripts/dmv-setup.sh .
   ```

   ```
   sh dmv-setup.sh
   ```

   Une fois l'opération terminée, vous devriez voir le message suivant s'afficher dans le terminal :

   ```
   ** DMV Sample App setup completed , enjoy!! **
   ```

5. Prenez le temps de parcourir l'exemple de code d'application dansAWS Cloud9, en particulier dans le chemin de répertoire suivant :`src/main/java/software/amazon/qldb/tutorial`.

Partie 3 : Exécuter l'exemple d'application QLDB DMV

Au cours de cette étape, vous apprendrez à exécuter les exemples de tâches d'application Amazon QLDB DMV à l'aide deAWS Cloud9. Pour exécuter l'exemple de code, retournez sur votreAWS Cloud9 terminal ou créez une nouvelle fenêtre de terminal comme vous l'avez fait dans la partie 2 : Configuration de votre IDE.

Pour exécuter l'exemple d'application

1. Pour accéder au répertoire racine du projet, exécutez la commande suivante dans votre terminal :

   ```
   cd ~/environment/amazon-qldb-dmv-sample-java
   ```

   Assurez-vous d'exécuter les exemples dans le chemin de répertoire suivant.

   ```
   /home/ec2-user/environment/amazon-qldb-dmv-sample-java/
   ```

2. La commande suivante montre la syntaxe Gradle pour exécuter chaque tâche.

   ```
   ./gradlew run -Dtutorial=Task
   ```

   Par exemple, exécutez la commande suivante pour répertorier tous les registres de votreCompte AWS et de votre région actuelle.

   ```
   ./gradlew run -Dtutorial=ListLedgers
   ```

3. PassezÉtape 1 : Créer un registre à démarrer le didacticiel et à créer un registre.
4. (Facultatif) Une fois le didacticiel terminé, nettoyez vosAWS CloudFormation ressources si vous n'en avez plus besoin.

   a. Ouvrez laAWS CloudFormation console à l'adresse https://console.aws.amazon.com/cloudformation et supprimez la pile que vous avez créée dans la partie 1 : provisionner vos ressources.

   b. Supprimez également laAWS Cloud9 pile que leAWS CloudFormation modèle a créée pour vous.

Étape 1 : Créer un registre

Au cours de cette étape, vous allez créer un nouveau registre Amazon QLDB nommé`vehicle-registration`.

Pour créer un registre

1. Consultez le fichier suivant (`Constants.java`), qui contient des valeurs constantes utilisées par tous les autres programmes de ce didacticiel.

   2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazon.ion.IonSystem;
import com.amazon.ion.system.IonSystemBuilder;
import com.fasterxml.jackson.databind.SerializationFeature;
```

```
import com.fasterxml.jackson.dataformat.ion.IonObjectMapper;
import com.fasterxml.jackson.dataformat.ion.ionvalue.IonValueMapper;

/**
 * Constant values used throughout this tutorial.
 */
public final class Constants {
    public static final int RETRY_LIMIT = 4;
    public static final String LEDGER_NAME = "vehicle-registration";
    public static final String STREAM_NAME = "vehicle-registration-stream";
    public static final String VEHICLE_REGISTRATION_TABLE_NAME =
 "VehicleRegistration";
    public static final String VEHICLE_TABLE_NAME = "Vehicle";
    public static final String PERSON_TABLE_NAME = "Person";
    public static final String DRIVERS_LICENSE_TABLE_NAME = "DriversLicense";
    public static final String VIN_INDEX_NAME = "VIN";
    public static final String PERSON_GOV_ID_INDEX_NAME = "GovId";
    public static final String
 VEHICLE_REGISTRATION_LICENSE_PLATE_NUMBER_INDEX_NAME = "LicensePlateNumber";
    public static final String DRIVER_LICENSE_NUMBER_INDEX_NAME =
 "LicenseNumber";
    public static final String DRIVER_LICENSE_PERSONID_INDEX_NAME = "PersonId";
    public static final String JOURNAL_EXPORT_S3_BUCKET_NAME_PREFIX = "qldb-
tutorial-journal-export";
    public static final String USER_TABLES = "information_schema.user_tables";
    public static final String LEDGER_NAME_WITH_TAGS = "tags";
    public static final IonSystem SYSTEM = IonSystemBuilder.standard().build();
    public static final IonObjectMapper MAPPER = new IonValueMapper(SYSTEM);

    private Constants() { }

    static {
        MAPPER.disable(SerializationFeature.WRITE_DATES_AS_TIMESTAMPS);
    }
}
```

1.x

> **Important**
>
> Pour le package Amazon Ion, vous devez utiliser l'espace de nomscom.amazon.ion dans votre application. AWS SDK for JavaCela dépend d'un

autre package Ion dans l'espace de nomssoftware.amazon.ion, mais il s'agit d'un package existant qui n'est pas compatible avec le pilote QLDB.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazon.ion.IonSystem;
import com.amazon.ion.system.IonSystemBuilder;
import com.fasterxml.jackson.databind.SerializationFeature;
import com.fasterxml.jackson.dataformat.ion.IonObjectMapper;
import com.fasterxml.jackson.dataformat.ion.ionvalue.IonValueMapper;

/**
 * Constant values used throughout this tutorial.
 */
public final class Constants {
```

```
    public static final int RETRY_LIMIT = 4;
    public static final String LEDGER_NAME = "vehicle-registration";
    public static final String STREAM_NAME = "vehicle-registration-stream";
    public static final String VEHICLE_REGISTRATION_TABLE_NAME =
 "VehicleRegistration";
    public static final String VEHICLE_TABLE_NAME = "Vehicle";
    public static final String PERSON_TABLE_NAME = "Person";
    public static final String DRIVERS_LICENSE_TABLE_NAME = "DriversLicense";
    public static final String VIN_INDEX_NAME = "VIN";
    public static final String PERSON_GOV_ID_INDEX_NAME = "GovId";
    public static final String
 VEHICLE_REGISTRATION_LICENSE_PLATE_NUMBER_INDEX_NAME = "LicensePlateNumber";
    public static final String DRIVER_LICENSE_NUMBER_INDEX_NAME =
 "LicenseNumber";
    public static final String DRIVER_LICENSE_PERSONID_INDEX_NAME = "PersonId";
    public static final String JOURNAL_EXPORT_S3_BUCKET_NAME_PREFIX = "qldb-
tutorial-journal-export";
    public static final String USER_TABLES = "information_schema.user_tables";
    public static final String LEDGER_NAME_WITH_TAGS = "tags";
    public static final IonSystem SYSTEM = IonSystemBuilder.standard().build();
    public static final IonObjectMapper MAPPER = new IonValueMapper(SYSTEM);

    private Constants() { }

    static {
        MAPPER.disable(SerializationFeature.WRITE_DATES_AS_TIMESTAMPS);
    }
}
```

> **Note**
>
> Cette`Constants` classe inclut une instance de la`IonValueMapper` classe open source Jackson. Vous pouvez utiliser ce mappeur pour traiter vos données Amazon Ion lorsque vous effectuez des transactions en lecture et en écriture.

Le`CreateLedger.java` fichier dépend également du programme suivant (`DescribeLedger.java`), qui décrit l'état actuel de votre registre.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazonaws.services.qldb.AmazonQLDB;
import com.amazonaws.services.qldb.model.DescribeLedgerRequest;
import com.amazonaws.services.qldb.model.DescribeLedgerResult;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
 * Describe a QLDB ledger.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class DescribeLedger {
    public static AmazonQLDB client = CreateLedger.getClient();
    public static final Logger log = LoggerFactory.getLogger(DescribeLedger.class);

    private DescribeLedger() { }
```

```
    public static void main(final String... args) {
        try {

            describe(Constants.LEDGER_NAME);

        } catch (Exception e) {
            log.error("Unable to describe a ledger!", e);
        }
    }

    /**
     * Describe a ledger.
     *
     * @param name
     *              Name of the ledger to describe.
     * @return {@link DescribeLedgerResult} from QLDB.
     */
    public static DescribeLedgerResult describe(final String name) {
        log.info("Let's describe ledger with name: {}...", name);
        DescribeLedgerRequest request = new DescribeLedgerRequest().withName(name);
        DescribeLedgerResult result = client.describeLedger(request);
        log.info("Success. Ledger description: {}", result);
        return result;
    }
}
```

2. Compilez et exécutez le`CreateLedger.java` programme pour créer un registre nommé`vehicle-registration`.

   2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
```

```
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.services.qldb.AmazonQLDB;
import com.amazonaws.services.qldb.AmazonQLDBClientBuilder;
import com.amazonaws.services.qldb.model.CreateLedgerRequest;
import com.amazonaws.services.qldb.model.CreateLedgerResult;
import com.amazonaws.services.qldb.model.DescribeLedgerResult;
import com.amazonaws.services.qldb.model.LedgerState;
import com.amazonaws.services.qldb.model.PermissionsMode;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
 * Create a ledger and wait for it to be active.
 * <p>
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class CreateLedger {
    public static final Logger log =
 LoggerFactory.getLogger(CreateLedger.class);
    public static final Long LEDGER_CREATION_POLL_PERIOD_MS = 10_000L;
    public static String endpoint = null;
    public static String region = null;
    public static AmazonQLDB client = getClient();
```

```
    private CreateLedger() {
    }

    /**
     * Build a low-level QLDB client.
     *
     * @return {@link AmazonQLDB} control plane client.
     */
    public static AmazonQLDB getClient() {
        AmazonQLDBClientBuilder builder = AmazonQLDBClientBuilder.standard();
        if (null != endpoint && null != region) {
            builder.setEndpointConfiguration(new
AwsClientBuilder.EndpointConfiguration(endpoint, region));
        }
        return builder.build();
    }

    public static void main(final String... args) throws Exception {
        try {
            client = getClient();

            create(Constants.LEDGER_NAME);

            waitForActive(Constants.LEDGER_NAME);

        } catch (Exception e) {
            log.error("Unable to create the ledger!", e);
            throw e;
        }
    }

    /**
     * Create a new ledger with the specified ledger name.
     *
     * @param ledgerName Name of the ledger to be created.
     * @return {@link CreateLedgerResult} from QLDB.
     */
    public static CreateLedgerResult create(final String ledgerName) {
        log.info("Let's create the ledger with name: {}...", ledgerName);
        CreateLedgerRequest request = new CreateLedgerRequest()
                .withName(ledgerName)
                .withPermissionsMode(PermissionsMode.ALLOW_ALL);
        CreateLedgerResult result = client.createLedger(request);
        log.info("Success. Ledger state: {}.", result.getState());
```

```
        return result;
    }

    /**
     * Wait for a newly created ledger to become active.
     *
     * @param ledgerName Name of the ledger to wait on.
     * @return {@link DescribeLedgerResult} from QLDB.
     * @throws InterruptedException if thread is being interrupted.
     */
    public static DescribeLedgerResult waitForActive(final String ledgerName)
 throws InterruptedException {
        log.info("Waiting for ledger to become active...");
        while (true) {
            DescribeLedgerResult result = DescribeLedger.describe(ledgerName);
            if (result.getState().equals(LedgerState.ACTIVE.name())) {
                log.info("Success. Ledger is active and ready to use.");
                return result;
            }
            log.info("The ledger is still creating. Please wait...");
            Thread.sleep(LEDGER_CREATION_POLL_PERIOD_MS);
        }
    }
}
```

1.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
```

```
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazonaws.services.qldb.AmazonQLDB;
import com.amazonaws.services.qldb.AmazonQLDBClientBuilder;
import com.amazonaws.services.qldb.model.CreateLedgerRequest;
import com.amazonaws.services.qldb.model.CreateLedgerResult;
import com.amazonaws.services.qldb.model.DescribeLedgerResult;
import com.amazonaws.services.qldb.model.LedgerState;
import com.amazonaws.services.qldb.model.PermissionsMode;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
 * Create a ledger and wait for it to be active.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class CreateLedger {
    public static final Logger log =
 LoggerFactory.getLogger(CreateLedger.class);
    public static final Long LEDGER_CREATION_POLL_PERIOD_MS = 10_000L;
    public static AmazonQLDB client = getClient();

    private CreateLedger() { }

    /**
     * Build a low-level QLDB client.
     *
     * @return {@link AmazonQLDB} control plane client.
     */
    public static AmazonQLDB getClient() {
```

```
        return AmazonQLDBClientBuilder.standard().build();
    }

    public static void main(final String... args) throws Exception {
        try {

            create(Constants.LEDGER_NAME);

            waitForActive(Constants.LEDGER_NAME);

        } catch (Exception e) {
            log.error("Unable to create the ledger!", e);
            throw e;
        }
    }

    /**
     * Create a new ledger with the specified ledger name.
     *
     * @param ledgerName
     *              Name of the ledger to be created.
     * @return {@link CreateLedgerResult} from QLDB.
     */
    public static CreateLedgerResult create(final String ledgerName) {
        log.info("Let's create the ledger with name: {}...", ledgerName);
        CreateLedgerRequest request = new CreateLedgerRequest()
                .withName(ledgerName)
                .withPermissionsMode(PermissionsMode.ALLOW_ALL);
        CreateLedgerResult result = client.createLedger(request);
        log.info("Success. Ledger state: {}.", result.getState());
        return result;
    }

    /**
     * Wait for a newly created ledger to become active.
     *
     * @param ledgerName
     *              Name of the ledger to wait on.
     * @return {@link DescribeLedgerResult} from QLDB.
     * @throws InterruptedException if thread is being interrupted.
     */
    public static DescribeLedgerResult waitForActive(final String ledgerName)
throws InterruptedException {
        log.info("Waiting for ledger to become active...");
```

```
        while (true) {
            DescribeLedgerResult result = DescribeLedger.describe(ledgerName);
            if (result.getState().equals(LedgerState.ACTIVE.name())) {
                log.info("Success. Ledger is active and ready to use.");
                return result;
            }
            log.info("The ledger is still creating. Please wait...");
            Thread.sleep(LEDGER_CREATION_POLL_PERIOD_MS);
        }
    }
}
```

> **Note**
>
> - Lors de l'`createLedger`appel, vous devez spécifier un nom de registre et un mode d'autorisation. Nous vous recommandons d'utiliser le mode`STANDARD` autorisations pour optimiser la sécurité des données de votre registre.
> - Lorsque vous créez un registre, la protection contre la suppression est activée par défaut. Il s'agit d'une fonctionnalité de QLDB qui empêche la suppression de registre par n'importe quel utilisateur. Vous avez la possibilité de désactiver la protection contre la suppression lors de la création d'un registre à l'aide de l'API QLDB ou duAWS Command Line Interface (AWS CLI).
> - Vous pouvez également spécifier des étiquettes à joindre à votre registre.

Pour vérifier votre connexion au nouveau registre, passez àÉtape 2 : Tester de la connectivité au registre.

Étape 2 : Tester de la connectivité au registre

Au cours de cette étape, vous devez vérifier que vous pouvez vous connecter au`vehicle-registration` registre dans Amazon QLDB à l'aide du point de terminaison de l'API de données transactionnelles.

Pour tester la connectivité au registre

1. Consultez le programme suivant (`ConnectToLedger.java`), qui crée une connexion de session de données au`vehicle-registration` registre.

2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import java.net.URI;
import java.net.URISyntaxException;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.services.qldbsession.QldbSessionClient;
import software.amazon.awssdk.services.qldbsession.QldbSessionClientBuilder;
import software.amazon.qldb.QldbDriver;
import software.amazon.qldb.RetryPolicy;

/**
 * Connect to a session for a given ledger using default settings.
 * <p>
```

```
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class ConnectToLedger {
    public static final Logger log =
 LoggerFactory.getLogger(ConnectToLedger.class);
    public static AwsCredentialsProvider credentialsProvider;
    public static String endpoint = null;
    public static String ledgerName = Constants.LEDGER_NAME;
    public static String region = null;
    public static QldbDriver driver;

    private ConnectToLedger() {
    }

    /**
     * Create a pooled driver for creating sessions.
     *
     * @param retryAttempts How many times the transaction will be retried in
     * case of a retryable issue happens like Optimistic Concurrency Control
 exception,
     * server side failures or network issues.
     * @return The pooled driver for creating sessions.
     */
    public static QldbDriver createQldbDriver(int retryAttempts) {
        QldbSessionClientBuilder builder = getAmazonQldbSessionClientBuilder();
        return QldbDriver.builder()
                             .ledger(ledgerName)
                             .transactionRetryPolicy(RetryPolicy
                                   .builder()
                                   .maxRetries(retryAttempts)
                                   .build())
                             .sessionClientBuilder(builder)
                             .build();
    }

    /**
     * Create a pooled driver for creating sessions.
     *
     * @return The pooled driver for creating sessions.
     */
    public static QldbDriver createQldbDriver() {
        QldbSessionClientBuilder builder = getAmazonQldbSessionClientBuilder();
```

```
        return QldbDriver.builder()
            .ledger(ledgerName)
            .transactionRetryPolicy(RetryPolicy.builder()

.maxRetries(Constants.RETRY_LIMIT).build())
            .sessionClientBuilder(builder)
            .build();
    }

    /**
     * Creates a QldbSession builder that is passed to the QldbDriver to connect
to the Ledger.
     *
     * @return An instance of the AmazonQLDBSessionClientBuilder
     */
    public static QldbSessionClientBuilder getAmazonQldbSessionClientBuilder() {
        QldbSessionClientBuilder builder = QldbSessionClient.builder();
        if (null != endpoint && null != region) {
            try {
                builder.endpointOverride(new URI(endpoint));
            } catch (URISyntaxException e) {
                throw new IllegalArgumentException(e);
            }
        }
        if (null != credentialsProvider) {
            builder.credentialsProvider(credentialsProvider);
        }
        return builder;
    }

    /**
     * Create a pooled driver for creating sessions.
     *
     * @return The pooled driver for creating sessions.
     */
    public static QldbDriver getDriver() {
        if (driver == null) {
            driver = createQldbDriver();
        }
        return driver;
    }


    public static void main(final String... args) {
```

```
        Iterable<String> tables = ConnectToLedger.getDriver().getTableNames();
        log.info("Existing tables in the ledger:");
        for (String table : tables) {
            log.info("- {} ", table);
        }
    }
}
```

> **Note**
>
> - Pour exécuter des opérations de données sur votre registre, vous devez créer une instance de la`QldbDriver` classe afin de vous connecter à un registre spécifique. Il s'agit d'un objet client différent de`AmazonQLDB` celui que vous avez utilisé à l'étape précédente pour créer le registre. Ce client précédent est uniquement utilisé pour exécuter les opérations d'API de gestion répertoriées dans leRéférence d'API Amazon QLDB.
> - Tout d'abord, créez un`QldbDriver` objet. Vous devez spécifier un nom de registre lorsque vous créez ce pilote.
>
>   Vous pouvez ensuite utiliser la`execute` méthode de ce pilote pour exécuter des instructions PartiQL.
> - Vous pouvez éventuellement spécifier un nombre maximum de nouvelles tentatives pour les exceptions de transaction. La`execute` méthode réessaie automatiquement les conflits de contrôle optimiste de la concurrence (OCC) et les autres exceptions transitoires courantes jusqu'à cette limite configurable. La valeur par défaut est 4.
>
>   Si la transaction échoue toujours une fois la limite atteinte, le pilote lance l'exception. Pour en savoir plus, consultez Comprendre la stratégie de nouvelle tentative avec le pilote dans Amazon QLDB.

1.x

```
/*
 * Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
```

```
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazonaws.auth.AWSCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.services.qldbsession.AmazonQLDBSessionClientBuilder;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.qldb.PooledQldbDriver;
import software.amazon.qldb.QldbDriver;
import software.amazon.qldb.QldbSession;
import software.amazon.qldb.exceptions.QldbClientException;

/**
 * Connect to a session for a given ledger using default settings.
 * <p>
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class ConnectToLedger {
    public static final Logger log =
 LoggerFactory.getLogger(ConnectToLedger.class);
```

```
    public static AWSCredentialsProvider credentialsProvider;
    public static String endpoint = null;
    public static String ledgerName = Constants.LEDGER_NAME;
    public static String region = null;
    private static PooledQldbDriver driver;

    private ConnectToLedger() {
    }

    /**
     * Create a pooled driver for creating sessions.
     *
     * @return The pooled driver for creating sessions.
     */
    public static PooledQldbDriver createQldbDriver() {
        AmazonQLDBSessionClientBuilder builder =
AmazonQLDBSessionClientBuilder.standard();
        if (null != endpoint && null != region) {
            builder.setEndpointConfiguration(new
AwsClientBuilder.EndpointConfiguration(endpoint, region));
        }
        if (null != credentialsProvider) {
            builder.setCredentials(credentialsProvider);
        }
        return PooledQldbDriver.builder()
                .withLedger(ledgerName)
                .withRetryLimit(Constants.RETRY_LIMIT)
                .withSessionClientBuilder(builder)
                .build();
    }

    /**
     * Create a pooled driver for creating sessions.
     *
     * @return The pooled driver for creating sessions.
     */
    public static PooledQldbDriver getDriver() {
        if (driver == null) {
            driver = createQldbDriver();
        }
        return driver;
    }

    /**
```

```
     * Connect to a ledger through a {@link QldbDriver}.
     *
     * @return {@link QldbSession}.
     */
    public static QldbSession createQldbSession() {
        return getDriver().getSession();
    }

    public static void main(final String... args) {
        try (QldbSession qldbSession = createQldbSession()) {
            log.info("Listing table names ");
            for (String tableName : qldbSession.getTableNames()) {
                log.info(tableName);
            }
        } catch (QldbClientException e) {
            log.error("Unable to create session.", e);
        }
    }
}
```

Note

- Pour exécuter des opérations de données sur votre registre, vous devez créer une instance de la`QldbDriver` classe`PooledQldbDriver` or pour vous connecter à un registre spécifique. Il s'agit d'un objet client différent de`AmazonQLDB` celui que vous avez utilisé à l'étape précédente pour créer le registre. Ce client précédent est uniquement utilisé pour exécuter les opérations d'API de gestion répertoriées dans leRéférence d'API Amazon QLDB.

  Nous vous recommandons d'utiliser,`PooledQldbDriver` sauf si vous devez implémenter un pool de sessions personnalisé avec`QldbDriver`. La taille du pool par défaut pour`PooledQldbDriver` est le nombre maximum de connexions HTTP ouvertes autorisées par le client de session.

- Tout d'abord, créez un`PooledQldbDriver` objet. Vous devez spécifier un nom de registre lorsque vous créez ce pilote.

  Vous pouvez ensuite utiliser la`execute` méthode de ce pilote pour exécuter des instructions PartiQL. Vous pouvez également créer manuellement une session à partir de cet objet pilote groupé et utiliser la`execute` méthode de la session. Une session représente une connexion unique avec le registre.

- Vous pouvez éventuellement spécifier un nombre maximum de nouvelles tentatives pour les exceptions de transaction. La`execute` méthode réessaie automatiquement les conflits de contrôle optimiste de la concurrence (OCC) et les autres exceptions transitoires courantes jusqu'à cette limite configurable. La valeur par défaut est 4.

  Si la transaction échoue toujours une fois la limite atteinte, le pilote lance l'exception. Pour en savoir plus, consultez Comprendre la stratégie de nouvelle tentative avec le pilote dans Amazon QLDB.

2. Compilez et exécutez le`ConnectToLedger.java` programme pour tester la connectivité de votre session de données au`vehicle-registration` registre.

En annulant leRégion AWS

L'exemple d'application se connecte à QLDB par défautRégion AWS, que vous pouvez définir comme décrit dans l'étape préalableConfiguration de vosAWS informations d'identification et de votre région par défaut. Vous pouvez également modifier la région en modifiant les propriétés du générateur de client de session QLDB.

2.x

L'exemple de code suivant instancie un nouvel`QldbSessionClientBuilder` objet.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.qldbsession.QldbSessionClientBuilder;

// This client builder will default to US East (Ohio)
QldbSessionClientBuilder builder = QldbSessionClient.builder()
    .region(Region.US_EAST_2);
```

Vous pouvez utiliser`region` cette méthode pour exécuter votre code sur QLDB dans n'importe quelle région où il est disponible. Pour obtenir la liste complète, consultez les points de terminaison et quotas Amazon QLDB dans le Références générales AWS.

1.x

L'exemple de code suivant instancie un nouvel`AmazonQLDBSessionClientBuilder` objet.

```
import com.amazonaws.regions.Regions;
```

```
import com.amazonaws.services.qldbsession.AmazonQLDBSessionClientBuilder;

// This client builder will default to US East (Ohio)
AmazonQLDBSessionClientBuilder builder = AmazonQLDBSessionClientBuilder.standard()
    .withRegion(Regions.US_EAST_2);
```

Vous pouvez utiliser`withRegion` cette méthode pour exécuter votre code sur QLDB dans n'importe quelle région où il est disponible. Pour obtenir la liste complète, consultez les points de terminaison et quotas Amazon QLDB dans le Références générales AWS.

Pour créer des tables dans le`vehicle-registration` grand livre, passez àÉtape 3 : Création de tables, d'index et d'exemples de données.

Étape 3 : Création de tables, d'index et d'exemples de données

Lorsque votre registre Amazon QLDB est actif et accepte les connexions, vous pouvez commencer à créer des tableaux contenant des données sur les véhicules, leurs propriétaires et leurs informations d'immatriculation. Après avoir créé les tables et les index, vous pouvez les charger avec des données.

Dans cette étape, vous allez créer quatre tables dans le`vehicle-registration` registre :

- `VehicleRegistration`
- `Vehicle`
- `Person`
- `DriversLicense`

Vous pouvez également créer les index suivants.

Nom de la table	Champ
`VehicleRegistration`	`VIN`
`VehicleRegistration`	`LicensePlateNumber`
`Vehicle`	`VIN`
`Person`	`GovId`

Nom de la table	Champ
`DriversLicense`	`LicenseNumber`
`DriversLicense`	`PersonId`

Lorsque vous insérez des exemples de données, vous insérez d'abord des documents dans le`Person` tableau. Ensuite, vous utilisez le système attribué`id` à chaque`Person` document pour remplir les champs correspondants dans les`DriversLicense` documents`VehicleRegistration` et les documents appropriés.

> **Tip**
>
> La meilleure pratique consiste à utiliser une clé étrangère attribuée par`id` le système à un document. Bien que vous puissiez définir des champs destinés à être des identifiants uniques (par exemple, le VIN d'un véhicule), le véritable identifiant unique d'un document est le sien`id`. Ce champ est inclus dans les métadonnées du document, que vous pouvez interroger dans la vue validée (la vue définie par le système d'une table).
> Pour plus d'informations sur les vues dans QLDB, consultezConcepts de base. Pour en savoir plus sur les métadonnées, consultezInterroger les métadonnées d'un document.

Pour configurer l'échantillon de données

1. Passez en revue les`.java` fichiers suivants. Ces classes de modèles représentent les documents que vous stockez dans les`vehicle-registration` tables. Ils sont sérialisables depuis et vers le format Amazon Ion.

   > **Note**
   >
   > Documents Amazon QLDBsont stockés au format Ion, qui est un sur-ensemble de JSON. Vous pouvez donc utiliser la bibliothèque FasterXML Jackson pour modéliser les données au format JSON.

   1. `DriversLicense.java`

```
/*
 * Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.model;

import com.fasterxml.jackson.annotation.JsonCreator;
import com.fasterxml.jackson.annotation.JsonProperty;
import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
import com.fasterxml.jackson.databind.annotation.JsonSerialize;
import software.amazon.qldb.tutorial.model.streams.RevisionData;

import java.time.LocalDate;

/**
 * Represents a driver's license, serializable to (and from) Ion.
 */
public final class DriversLicense implements RevisionData {
    private final String personId;
    private final String licenseNumber;
    private final String licenseType;
```

```
    @JsonSerialize(using = IonLocalDateSerializer.class)
    @JsonDeserialize(using = IonLocalDateDeserializer.class)
    private final LocalDate validFromDate;

    @JsonSerialize(using = IonLocalDateSerializer.class)
    @JsonDeserialize(using = IonLocalDateDeserializer.class)
    private final LocalDate validToDate;

    @JsonCreator
    public DriversLicense(@JsonProperty("PersonId") final String personId,
                          @JsonProperty("LicenseNumber") final String
licenseNumber,
                          @JsonProperty("LicenseType") final String licenseType,
                          @JsonProperty("ValidFromDate") final LocalDate
validFromDate,
                          @JsonProperty("ValidToDate") final LocalDate
validToDate) {
        this.personId = personId;
        this.licenseNumber = licenseNumber;
        this.licenseType = licenseType;
        this.validFromDate = validFromDate;
        this.validToDate = validToDate;
    }

    @JsonProperty("PersonId")
    public String getPersonId() {
        return personId;
    }

    @JsonProperty("LicenseNumber")
    public String getLicenseNumber() {
        return licenseNumber;
    }

    @JsonProperty("LicenseType")
    public String getLicenseType() {
        return licenseType;
    }

    @JsonProperty("ValidFromDate")
    public LocalDate getValidFromDate() {
        return  validFromDate;
    }
```

```
    @JsonProperty("ValidToDate")
    public LocalDate getValidToDate() {
        return  validToDate;
    }

    @Override
    public String toString() {
        return "DriversLicense{" +
                "personId='" + personId + '\'' +
                ", licenseNumber='" + licenseNumber + '\'' +
                ", licenseType='" + licenseType + '\'' +
                ", validFromDate=" + validFromDate +
                ", validToDate=" + validToDate +
                '}';
    }
}
```

2. Person.java

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */
```

```
package software.amazon.qldb.tutorial.model;

import java.time.LocalDate;

import com.fasterxml.jackson.annotation.JsonCreator;
import com.fasterxml.jackson.annotation.JsonProperty;
import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
import com.fasterxml.jackson.databind.annotation.JsonSerialize;

import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.Constants;
import software.amazon.qldb.tutorial.model.streams.RevisionData;

/**
 * Represents a person, serializable to (and from) Ion.
 */
public final class Person implements RevisionData {
    private final String firstName;
    private final String lastName;

    @JsonSerialize(using = IonLocalDateSerializer.class)
    @JsonDeserialize(using = IonLocalDateDeserializer.class)
    private final LocalDate dob;
    private final String govId;
    private final String govIdType;
    private final String address;

    @JsonCreator
    public Person(@JsonProperty("FirstName") final String firstName,
                  @JsonProperty("LastName") final String lastName,
                  @JsonProperty("DOB") final LocalDate dob,
                  @JsonProperty("GovId") final String govId,
                  @JsonProperty("GovIdType") final String govIdType,
                  @JsonProperty("Address") final String address) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.dob = dob;
        this.govId = govId;
        this.govIdType = govIdType;
        this.address = address;
    }

    @JsonProperty("Address")
    public String getAddress() {
```

```
        return address;
    }

    @JsonProperty("DOB")
    public LocalDate getDob() {
        return dob;
    }

    @JsonProperty("FirstName")
    public String getFirstName() {
        return firstName;
    }

    @JsonProperty("LastName")
    public String getLastName() {
        return lastName;
    }

    @JsonProperty("GovId")
    public String getGovId() {
        return govId;
    }

    @JsonProperty("GovIdType")
    public String getGovIdType() {
        return govIdType;
    }

    /**
     * This returns the unique document ID given a specific government ID.
     *
     * @param txn
     *              A transaction executor object.
     * @param govId
     *              The government ID of a driver.
     * @return the unique document ID.
     */
    public static String getDocumentIdByGovId(final TransactionExecutor txn,
final String govId) {
        return SampleData.getDocumentId(txn, Constants.PERSON_TABLE_NAME,
"GovId", govId);
    }

    @Override
```

```
    public String toString() {
        return "Person{" +
                "firstName='" + firstName + '\'' +
                ", lastName='" + lastName + '\'' +
                ", dob=" + dob +
                ", govId='" + govId + '\'' +
                ", govIdType='" + govIdType + '\'' +
                ", address='" + address + '\'' +
                '}';
    }
}
```

3. VehicleRegistration.java

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.model;

import com.fasterxml.jackson.annotation.JsonCreator;
import com.fasterxml.jackson.annotation.JsonProperty;
import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
```

```
import com.fasterxml.jackson.databind.annotation.JsonSerialize;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.Constants;
import software.amazon.qldb.tutorial.model.streams.RevisionData;

import java.math.BigDecimal;
import java.time.LocalDate;

/**
 * Represents a vehicle registration, serializable to (and from) Ion.
 */
public final class VehicleRegistration implements RevisionData {

    private final String vin;
    private final String licensePlateNumber;
    private final String state;
    private final String city;
    private final BigDecimal pendingPenaltyTicketAmount;
    private final LocalDate validFromDate;
    private final LocalDate validToDate;
    private final Owners owners;

    @JsonCreator
    public VehicleRegistration(@JsonProperty("VIN") final String vin,
                               @JsonProperty("LicensePlateNumber") final String
 licensePlateNumber,
                               @JsonProperty("State") final String state,
                               @JsonProperty("City") final String city,
                               @JsonProperty("PendingPenaltyTicketAmount") final
 BigDecimal pendingPenaltyTicketAmount,
                               @JsonProperty("ValidFromDate") final LocalDate
 validFromDate,
                               @JsonProperty("ValidToDate") final LocalDate
 validToDate,
                               @JsonProperty("Owners") final Owners owners) {
        this.vin = vin;
        this.licensePlateNumber = licensePlateNumber;
        this.state = state;
        this.city = city;
        this.pendingPenaltyTicketAmount = pendingPenaltyTicketAmount;
        this.validFromDate = validFromDate;
        this.validToDate = validToDate;
        this.owners = owners;
    }
```

```
    @JsonProperty("City")
    public String getCity() {
        return city;
    }

    @JsonProperty("LicensePlateNumber")
    public String getLicensePlateNumber() {
        return licensePlateNumber;
    }

    @JsonProperty("Owners")
    public Owners getOwners() {
        return owners;
    }

    @JsonProperty("PendingPenaltyTicketAmount")
    public BigDecimal getPendingPenaltyTicketAmount() {
        return pendingPenaltyTicketAmount;
    }

    @JsonProperty("State")
    public String getState() {
        return state;
    }

    @JsonProperty("ValidFromDate")
    @JsonSerialize(using = IonLocalDateSerializer.class)
    @JsonDeserialize(using = IonLocalDateDeserializer.class)
    public LocalDate getValidFromDate() {
        return validFromDate;
    }

    @JsonProperty("ValidToDate")
    @JsonSerialize(using = IonLocalDateSerializer.class)
    @JsonDeserialize(using = IonLocalDateDeserializer.class)
    public LocalDate getValidToDate() {
        return validToDate;
    }

    @JsonProperty("VIN")
    public String getVin() {
        return vin;
    }
```

```
    /**
     * Returns the unique document ID of a vehicle given a specific VIN.
     *
     * @param txn
     *              A transaction executor object.
     * @param vin
     *              The VIN of a vehicle.
     * @return the unique document ID of the specified vehicle.
     */
    public static String getDocumentIdByVin(final TransactionExecutor txn, final
 String vin) {
        return SampleData.getDocumentId(txn,
 Constants.VEHICLE_REGISTRATION_TABLE_NAME, "VIN", vin);
    }

    @Override
    public String toString() {
        return "VehicleRegistration{" +
                "vin='" + vin + '\'' +
                ", licensePlateNumber='" + licensePlateNumber + '\'' +
                ", state='" + state + '\'' +
                ", city='" + city + '\'' +
                ", pendingPenaltyTicketAmount=" + pendingPenaltyTicketAmount +
                ", validFromDate=" + validFromDate +
                ", validToDate=" + validToDate +
                ", owners=" + owners +
                '}';
    }
}
```

4. Vehicle.java

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
```

```
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.model;

import com.fasterxml.jackson.annotation.JsonCreator;
import com.fasterxml.jackson.annotation.JsonProperty;
import software.amazon.qldb.tutorial.model.streams.RevisionData;

/**
 * Represents a vehicle, serializable to (and from) Ion.
 */
public final class Vehicle implements RevisionData {
    private final String vin;
    private final String type;
    private final int year;
    private final String make;
    private final String model;
    private final String color;

    @JsonCreator
    public Vehicle(@JsonProperty("VIN") final String vin,
                   @JsonProperty("Type") final String type,
                   @JsonProperty("Year") final int year,
                   @JsonProperty("Make") final String make,
                   @JsonProperty("Model") final String model,
                   @JsonProperty("Color") final String color) {
        this.vin = vin;
        this.type = type;
        this.year = year;
        this.make = make;
        this.model = model;
```

```
        this.color = color;
    }

    @JsonProperty("Color")
    public String getColor() {
        return color;
    }

    @JsonProperty("Make")
    public String getMake() {
        return make;
    }

    @JsonProperty("Model")
    public String getModel() {
        return model;
    }

    @JsonProperty("Type")
    public String getType() {
        return type;
    }

    @JsonProperty("VIN")
    public String getVin() {
        return vin;
    }

    @JsonProperty("Year")
    public int getYear() {
        return year;
    }

    @Override
    public String toString() {
        return "Vehicle{" +
                "vin='" + vin + '\'' +
                ", type='" + type + '\'' +
                ", year=" + year +
                ", make='" + make + '\'' +
                ", model='" + model + '\'' +
                ", color='" + color + '\'' +
                '}';
    }
```

```
}
```

5. Owner.java

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.model;

import com.fasterxml.jackson.annotation.JsonProperty;

/**
 * Represents a vehicle owner, serializable to (and from) Ion.
 */
public final class Owner {
    private final String personId;

    public Owner(@JsonProperty("PersonId") final String personId) {
        this.personId = personId;
    }

    @JsonProperty("PersonId")
```

```
    public String getPersonId() {
        return personId;
    }

    @Override
    public String toString() {
        return "Owner{" +
                "personId='" + personId + '\'' +
                '}';
    }
}
```

6. Owners.java

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.model;

import com.fasterxml.jackson.annotation.JsonProperty;

import java.util.List;
```

```
/**
 * Represents a set of owners for a given vehicle, serializable to (and from)
 Ion.
 */
public final class Owners {
    private final Owner primaryOwner;
    private final List<Owner> secondaryOwners;

    public Owners(@JsonProperty("PrimaryOwner") final Owner primaryOwner,
                  @JsonProperty("SecondaryOwners") final List<Owner>
 secondaryOwners) {
        this.primaryOwner = primaryOwner;
        this.secondaryOwners = secondaryOwners;
    }

    @JsonProperty("PrimaryOwner")
    public Owner getPrimaryOwner() {
        return primaryOwner;
    }

    @JsonProperty("SecondaryOwners")
    public List<Owner> getSecondaryOwners() {
        return secondaryOwners;
    }

    @Override
    public String toString() {
        return "Owners{" +
                "primaryOwner=" + primaryOwner +
                ", secondaryOwners=" + secondaryOwners +
                '}';
    }
}
```

7. DmlResultDocument.java

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
```

```
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.qldb;

import com.fasterxml.jackson.annotation.JsonCreator;
import com.fasterxml.jackson.annotation.JsonProperty;

/**
 * Contains information about an individual document inserted or modified
 * as a result of DML.
 */
public class DmlResultDocument {

    private String documentId;

    @JsonCreator
    public DmlResultDocument(@JsonProperty("documentId") final String documentId)
 {
        this.documentId = documentId;
    }

    public String getDocumentId() {
        return documentId;
    }

    @Override
    public String toString() {
```

```
        return "DmlResultDocument{"
            + "documentId='" + documentId + '\''
            + '}';
    }
}
```

8. RevisionData.java

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.model.streams;

/**
 * Allows modeling the content of all revisions as a generic revision data. Used
 * in the {@link Revision} and extended by domain models in {@link
 * software.amazon.qldb.tutorial.model} to make it easier to write the {@link
 * Revision.RevisionDataDeserializer} that must deserialize the {@link
 * Revision#data} from different domain models.
 */
public interface RevisionData { }
```

9. RevisionMetadata.java

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.qldb;

import com.amazon.ion.IonInt;
import com.amazon.ion.IonString;
import com.amazon.ion.IonStruct;
import com.amazon.ion.IonTimestamp;
import com.fasterxml.jackson.annotation.JsonCreator;
import com.fasterxml.jackson.annotation.JsonProperty;
import com.fasterxml.jackson.databind.annotation.JsonSerialize;
import com.fasterxml.jackson.dataformat.ion.IonTimestampSerializers;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.util.Date;
import java.util.Objects;
```

```
/**
 * Represents the metadata field of a QLDB Document
 */
public class RevisionMetadata {
    private static final Logger log =
 LoggerFactory.getLogger(RevisionMetadata.class);
    private final String id;
    private final long version;
    @JsonSerialize(using =
 IonTimestampSerializers.IonTimestampJavaDateSerializer.class)
    private final Date txTime;
    private final String txId;

    @JsonCreator
    public RevisionMetadata(@JsonProperty("id") final String id,
                            @JsonProperty("version") final long version,
                            @JsonProperty("txTime") final Date txTime,
                            @JsonProperty("txId") final String txId) {
        this.id = id;
        this.version = version;
        this.txTime = txTime;
        this.txId = txId;
    }

    /**
     * Gets the unique ID of a QLDB document.
     *
     * @return the document ID.
     */
    public String getId() {
        return id;
    }

    /**
     * Gets the version number of the document in the document's modification
 history.
     * @return the version number.
     */
    public long getVersion() {
        return version;
    }

    /**
     * Gets the time during which the document was modified.
```

```
     *
     * @return the transaction time.
     */
    public Date getTxTime() {
        return txTime;
    }

    /**
     * Gets the transaction ID associated with this document.
     *
     * @return the transaction ID.
     */
    public String getTxId() {
        return txId;
    }

    public static RevisionMetadata fromIon(final IonStruct ionStruct) {
        if (ionStruct == null) {
            throw new IllegalArgumentException("Metadata cannot be null");
        }
        try {
            IonString id = (IonString) ionStruct.get("id");
            IonInt version = (IonInt) ionStruct.get("version");
            IonTimestamp txTime = (IonTimestamp) ionStruct.get("txTime");
            IonString txId = (IonString) ionStruct.get("txId");
            if (id == null || version == null || txTime == null || txId == null)
{
                throw new IllegalArgumentException("Document is missing required
fields");
            }
            return new RevisionMetadata(id.stringValue(), version.longValue(),
new Date(txTime.getMillis()), txId.stringValue());
        } catch (ClassCastException e) {
            log.error("Failed to parse ion document");
            throw new IllegalArgumentException("Document members are not of the
correct type", e);
        }
    }

    /**
     * Converts a {@link RevisionMetadata} object to a string.
     *
     * @return the string representation of the {@link QldbRevision} object.
     */
```

```
    @Override
    public String toString() {
        return "Metadata{"
                + "id='" + id + '\''
                + ", version=" + version
                + ", txTime=" + txTime
                + ", txId='" + txId
                + '\''
                + '}';
    }

    /**
     * Check whether two {@link RevisionMetadata} objects are equivalent.
     *
     * @return {@code true} if the two objects are equal, {@code false}
 otherwise.
     */
    @Override
    public boolean equals(Object o) {
        if (this == o) { return true; }
        if (o == null || getClass() != o.getClass()) { return false; }
        RevisionMetadata metadata = (RevisionMetadata) o;
        return version == metadata.version
                && id.equals(metadata.id)
                && txTime.equals(metadata.txTime)
                && txId.equals(metadata.txId);
    }

    /**
     * Generate a hash code for the {@link RevisionMetadata} object.
     *
     * @return the hash code.
     */
    @Override
    public int hashCode() {
        // CHECKSTYLE:OFF - Disabling as we are generating a hashCode of multiple
 properties.
        return Objects.hash(id, version, txTime, txId);
        // CHECKSTYLE:ON
    }
}
```

10QldbRevision.java

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.qldb;

import com.amazon.ion.IonBlob;
import com.amazon.ion.IonStruct;
import com.fasterxml.jackson.annotation.JsonCreator;
import com.fasterxml.jackson.annotation.JsonProperty;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.qldb.tutorial.Constants;
import software.amazon.qldb.tutorial.Verifier;

import java.io.IOException;
import java.util.Arrays;
import java.util.Objects;

/**
 * Represents a QldbRevision including both user data and metadata.
 */
```

```
public final class QldbRevision {
    private static final Logger log =
 LoggerFactory.getLogger(QldbRevision.class);

    private final BlockAddress blockAddress;
    private final RevisionMetadata metadata;
    private final byte[] hash;
    private final byte[] dataHash;
    private final IonStruct data;

    @JsonCreator
    public QldbRevision(@JsonProperty("blockAddress") final BlockAddress
 blockAddress,
                        @JsonProperty("metadata") final RevisionMetadata
 metadata,
                        @JsonProperty("hash") final byte[] hash,
                        @JsonProperty("dataHash") final byte[] dataHash,
                        @JsonProperty("data") final IonStruct data) {
        this.blockAddress = blockAddress;
        this.metadata = metadata;
        this.hash = hash;
        this.dataHash = dataHash;
        this.data = data;
    }

    /**
     * Gets the unique ID of a QLDB document.
     *
     * @return the {@link BlockAddress} object.
     */
    public BlockAddress getBlockAddress() {
        return blockAddress;
    }

    /**
     * Gets the metadata of the revision.
     *
     * @return the {@link RevisionMetadata} object.
     */
    public RevisionMetadata getMetadata() {
        return metadata;
    }

    /**
```

```
     * Gets the SHA-256 hash value of the revision.
     * This is equivalent to the hash of the revision metadata and data.
     *
     * @return the byte array representing the hash.
     */
    public byte[] getHash() {
        return hash;
    }

    /**
     * Gets the SHA-256 hash value of the data portion of the revision.
     * This is only present if the revision is redacted.
     *
     * @return the byte array representing the hash.
     */
    public byte[] getDataHash() {
        return dataHash;
    }

    /**
     * Gets the revision data.
     *
     * @return the revision data.
     */
    public IonStruct getData() {
        return data;
    }

    /**
     * Returns true if the revision has been redacted.
     * @return a boolean value representing the redaction status
     * of this revision.
     */
    public Boolean isRedacted() {
        return dataHash != null;
    }

    /**
     * Constructs a new {@link QldbRevision} from an {@link IonStruct}.
     *
     * The specified {@link IonStruct} must include the following fields
     *
     * - blockAddress -- a {@link BlockAddress},
     * - metadata -- a {@link RevisionMetadata},
```

```
    * - hash -- the revision's hash calculated by QLDB,
    * - dataHash -- the user data's hash calculated by QLDB (only present if
revision is redacted),
    * - data -- an {@link IonStruct} containing user data in the document.
    *
    * If any of these fields are missing or are malformed, then throws {@link
IllegalArgumentException}.
    *
    * If the document hash calculated from the members of the specified {@link
IonStruct} does not match
    * the hash member of the {@link IonStruct} then throws {@link
IllegalArgumentException}.
    *
    * @param ionStruct
    *              The {@link IonStruct} that contains a {@link QldbRevision}
object.
    * @return the converted {@link QldbRevision} object.
    * @throws IOException if failed to parse parameter {@link IonStruct}.
    */
   public static QldbRevision fromIon(final IonStruct ionStruct) throws
IOException {
       try {
           BlockAddress blockAddress =
Constants.MAPPER.readValue(ionStruct.get("blockAddress"), BlockAddress.class);
           IonBlob revisionHash = (IonBlob) ionStruct.get("hash");
           IonStruct metadataStruct = (IonStruct) ionStruct.get("metadata");
           IonStruct data = ionStruct.get("data") == null ||
ionStruct.get("data").isNullValue() ?
               null : (IonStruct) ionStruct.get("data");
           IonBlob dataHash = ionStruct.get("dataHash") == null ||
ionStruct.get("dataHash").isNullValue() ?
               null : (IonBlob) ionStruct.get("dataHash");
           if (revisionHash == null || metadataStruct == null) {
               throw new IllegalArgumentException("Document is missing required
fields");
           }
           byte[] dataHashBytes = dataHash != null ? dataHash.getBytes() :
QldbIonUtils.hashIonValue(data);
           verifyRevisionHash(metadataStruct, dataHashBytes,
revisionHash.getBytes());
           RevisionMetadata metadata = RevisionMetadata.fromIon(metadataStruct);
           return new QldbRevision(
                   blockAddress,
                   metadata,
```

```
                    revisionHash.getBytes(),
                    dataHash != null ? dataHash.getBytes() : null,
                    data
            );
        } catch (ClassCastException e) {
            log.error("Failed to parse ion document");
            throw new IllegalArgumentException("Document members are not of the
correct type", e);
        }
    }

    /**
     * Converts a {@link QldbRevision} object to string.
     *
     * @return the string representation of the {@link QldbRevision} object.
     */
    @Override
    public String toString() {
        return "QldbRevision{" +
                "blockAddress=" + blockAddress +
                ", metadata=" + metadata +
                ", hash=" + Arrays.toString(hash) +
                ", dataHash=" + Arrays.toString(dataHash) +
                ", data=" + data +
                '}';
    }

    /**
     * Check whether two {@link QldbRevision} objects are equivalent.
     *
     * @return {@code true} if the two objects are equal, {@code false}
otherwise.
     */
    @Override
    public boolean equals(final Object o) {
        if (this == o) {
            return true;
        }
        if (!(o instanceof QldbRevision)) {
            return false;
        }
        final QldbRevision that = (QldbRevision) o;
        return Objects.equals(getBlockAddress(), that.getBlockAddress())
                && Objects.equals(getMetadata(), that.getMetadata())
```

```
                && Arrays.equals(getHash(), that.getHash())
                && Arrays.equals(getDataHash(), that.getDataHash())
                && Objects.equals(getData(), that.getData());
    }

    /**
     * Create a hash code for the {@link QldbRevision} object.
     *
     * @return the hash code.
     */
    @Override
    public int hashCode() {
        // CHECKSTYLE:OFF - Disabling as we are generating a hashCode of multiple
properties.
        int result = Objects.hash(blockAddress, metadata, data);
        // CHECKSTYLE:ON
        result = 31 * result + Arrays.hashCode(hash);
        return result;
    }

    /**
     * Throws an IllegalArgumentException if the hash of the revision data and
metadata
     * does not match the hash provided by QLDB with the revision.
     */
    public void verifyRevisionHash() {
        // Certain internal-only system revisions only contain a hash which
cannot be
        // further computed. However, these system hashes still participate to
validate
        // the journal block. User revisions will always contain values for all
fields
        // and can therefore have their hash computed.
        if (blockAddress == null && metadata == null && data == null && dataHash
== null) {
            return;
        }

        try {
            IonStruct metadataIon = (IonStruct)
Constants.MAPPER.writeValueAsIonValue(metadata);
            byte[] dataHashBytes = isRedacted() ? dataHash :
QldbIonUtils.hashIonValue(data);
            verifyRevisionHash(metadataIon, dataHashBytes, hash);
```

```
        } catch (IOException e) {
            throw new IllegalArgumentException("Could not encode revision
 metadata to ion.", e);
        }
    }

    private static void verifyRevisionHash(IonStruct metadata, byte[] dataHash,
 byte[] expectedHash) {
        byte[] metadataHash = QldbIonUtils.hashIonValue(metadata);
        byte[] candidateHash = Verifier.dot(metadataHash, dataHash);
        if (!Arrays.equals(candidateHash, expectedHash)) {
            throw new IllegalArgumentException("Hash entry of QLDB revision and
 computed hash "
                    + "of QLDB revision do not match");
        }
    }
}
```

11IonLocalDateDeserializer.java

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */
```

```
package software.amazon.qldb.tutorial.model;

import com.amazon.ion.Timestamp;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.DeserializationContext;
import com.fasterxml.jackson.databind.JsonDeserializer;

import java.io.IOException;
import java.time.LocalDate;

/**
 * Deserializes [java.time.LocalDate] from Ion.
 */
public class IonLocalDateDeserializer extends JsonDeserializer<LocalDate> {

    @Override
    public LocalDate deserialize(JsonParser jp, DeserializationContext ctxt)
 throws IOException {
        return timestampToLocalDate((Timestamp) jp.getEmbeddedObject());
    }

    private LocalDate timestampToLocalDate(Timestamp timestamp) {
        return LocalDate.of(timestamp.getYear(), timestamp.getMonth(),
 timestamp.getDay());
    }
}
```

12IonLocalDateSerializer.java

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
```

```
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.model;

import com.amazon.ion.Timestamp;
import com.fasterxml.jackson.core.JsonGenerator;
import com.fasterxml.jackson.databind.SerializerProvider;
import com.fasterxml.jackson.databind.ser.std.StdScalarSerializer;
import com.fasterxml.jackson.dataformat.ion.IonGenerator;

import java.io.IOException;
import java.time.LocalDate;

/**
 * Serializes [java.time.LocalDate] to Ion.
 */
public class IonLocalDateSerializer extends StdScalarSerializer<LocalDate> {

    public IonLocalDateSerializer() {
        super(LocalDate.class);
    }

    @Override
    public void serialize(LocalDate date, JsonGenerator jsonGenerator,
 SerializerProvider serializerProvider) throws IOException {
        Timestamp timestamp = Timestamp.forDay(date.getYear(),
 date.getMonthValue(), date.getDayOfMonth());
        ((IonGenerator) jsonGenerator).writeValue(timestamp);
    }
}
```

2. Consultez le fichier suivant (`SampleData.java`), qui représente les exemples de données que vous insérez dans les`vehicle-registration` tableaux.

2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.model;

import com.amazon.ion.IonString;
import com.amazon.ion.IonStruct;
import com.amazon.ion.IonValue;
import java.io.IOException;
import java.math.BigDecimal;
import java.text.ParseException;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;
import software.amazon.qldb.Result;
```

```
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.ConnectToLedger;
import software.amazon.qldb.tutorial.Constants;
import software.amazon.qldb.tutorial.qldb.DmlResultDocument;
import software.amazon.qldb.tutorial.qldb.QldbRevision;

/**
 * Sample domain objects for use throughout this tutorial.
 */
public final class SampleData {
    public static final DateTimeFormatter DATE_TIME_FORMAT =
 DateTimeFormatter.ofPattern("yyyy-MM-dd");

    public static final List<VehicleRegistration> REGISTRATIONS =
 Collections.unmodifiableList(Arrays.asList(
            new VehicleRegistration("1N4AL11D75C109151", "LEWISR261LL", "WA",
 "Seattle",
                    BigDecimal.valueOf(90.25), convertToLocalDate("2017-08-21"),
 convertToLocalDate("2020-05-11"),
                    new Owners(new Owner(null), Collections.emptyList())),
            new VehicleRegistration("KM8SRDHF6EU074761", "CA762X", "WA", "Kent",
                    BigDecimal.valueOf(130.75),
 convertToLocalDate("2017-09-14"), convertToLocalDate("2020-06-25"),
                    new Owners(new Owner(null), Collections.emptyList())),
            new VehicleRegistration("3HGGK5G53FM761765", "CD820Z", "WA",
 "Everett",
                    BigDecimal.valueOf(442.30),
 convertToLocalDate("2011-03-17"), convertToLocalDate("2021-03-24"),
                    new Owners(new Owner(null), Collections.emptyList())),
            new VehicleRegistration("1HVBBAANXWH544237", "LS477D", "WA",
 "Tacoma",
                    BigDecimal.valueOf(42.20), convertToLocalDate("2011-10-26"),
 convertToLocalDate("2023-09-25"),
                    new Owners(new Owner(null), Collections.emptyList())),
            new VehicleRegistration("1C4RJFAG0FC625797", "TH393F", "WA",
 "Olympia",
                    BigDecimal.valueOf(30.45), convertToLocalDate("2013-09-02"),
 convertToLocalDate("2024-03-19"),
                    new Owners(new Owner(null), Collections.emptyList()))
    ));

    public static final List<Vehicle> VEHICLES =
 Collections.unmodifiableList(Arrays.asList(
```

```
            new Vehicle("1N4AL11D75C109151", "Sedan", 2011,  "Audi", "A5",
"Silver"),
            new Vehicle("KM8SRDHF6EU074761", "Sedan", 2015, "Tesla", "Model S",
"Blue"),
            new Vehicle("3HGGK5G53FM761765", "Motorcycle", 2011, "Ducati",
"Monster 1200", "Yellow"),
            new Vehicle("1HVBBAANXWH544237", "Semi", 2009, "Ford", "F 150",
"Black"),
            new Vehicle("1C4RJFAG0FC625797", "Sedan", 2019, "Mercedes", "CLK
350", "White")
    ));

    public static final List<Person> PEOPLE =
Collections.unmodifiableList(Arrays.asList(
            new Person("Raul", "Lewis", convertToLocalDate("1963-08-19"),
                    "LEWISR261LL", "Driver License",  "1719 University Street,
Seattle, WA, 98109"),
            new Person("Brent", "Logan", convertToLocalDate("1967-07-03"),
                    "LOGANB486CG", "Driver License", "43 Stockert Hollow Road,
Everett, WA, 98203"),
            new Person("Alexis", "Pena", convertToLocalDate("1974-02-10"),
                    "744 849 301", "SSN", "4058 Melrose Street, Spokane Valley,
WA, 99206"),
            new Person("Melvin", "Parker", convertToLocalDate("1976-05-22"),
                    "P626-168-229-765", "Passport", "4362 Ryder Avenue, Seattle,
WA, 98101"),
            new Person("Salvatore", "Spencer", convertToLocalDate("1997-11-15"),
                    "S152-780-97-415-0", "Passport", "4450 Honeysuckle Lane,
Seattle, WA, 98101")
    ));

    public static final List<DriversLicense> LICENSES =
Collections.unmodifiableList(Arrays.asList(
            new DriversLicense(null, "LEWISR261LL", "Learner",
                    convertToLocalDate("2016-12-20"),
convertToLocalDate("2020-11-15")),
            new DriversLicense(null, "LOGANB486CG", "Probationary",
                    convertToLocalDate("2016-04-06"),
convertToLocalDate("2020-11-15")),
            new DriversLicense(null, "744 849 301", "Full",
                    convertToLocalDate("2017-12-06"),
convertToLocalDate("2022-10-15")),
            new DriversLicense(null, "P626-168-229-765", "Learner",
```

```
                    convertToLocalDate("2017-08-16"),
convertToLocalDate("2021-11-15")),
            new DriversLicense(null, "S152-780-97-415-0", "Probationary",
                    convertToLocalDate("2015-08-15"),
convertToLocalDate("2021-08-21"))
    ));

    private SampleData() { }

    /**
     * Converts a date string with the format 'yyyy-MM-dd' into a {@link
java.util.Date} object.
     *
     * @param date
     *              The date string to convert.
     * @return {@link java.time.LocalDate} or null if there is a {@link
ParseException}
     */
    public static synchronized LocalDate convertToLocalDate(String date) {
        return LocalDate.parse(date, DATE_TIME_FORMAT);
    }

    /**
     * Convert the result set into a list of IonValues.
     *
     * @param result
     *              The result set to convert.
     * @return a list of IonValues.
     */
    public static List<IonValue> toIonValues(Result result) {
        final List<IonValue> valueList = new ArrayList<>();
        result.iterator().forEachRemaining(valueList::add);
        return valueList;
    }

    /**
     * Get the document ID of a particular document.
     *
     * @param txn
     *              A transaction executor object.
     * @param tableName
     *              Name of the table containing the document.
     * @param identifier
     *              The identifier used to narrow down the search.
```

```
    * @param value
    *              Value of the identifier.
    * @return the list of document IDs in the result set.
    */
   public static String getDocumentId(final TransactionExecutor txn, final
String tableName,
                                      final String identifier, final String
value) {
       try {
           final List<IonValue> parameters =
Collections.singletonList(Constants.MAPPER.writeValueAsIonValue(value));
           final String query = String.format("SELECT metadata.id FROM
_ql_committed_%s AS p WHERE p.data.%s = ?",
                   tableName, identifier);
           Result result = txn.execute(query, parameters);
           if (result.isEmpty()) {
               throw new IllegalStateException("Unable to retrieve document ID
using " + value);
           }
           return getStringValueOfStructField((IonStruct)
result.iterator().next(), "id");
       } catch (IOException ioe) {
           throw new IllegalStateException(ioe);
       }
   }

   /**
    * Get the document by ID.
    *
    * @param tableName
    *              Name of the table to insert documents into.
    * @param documentId
    *              The unique ID of a document in the Person table.
    * @return a {@link QldbRevision} object.
    * @throws IllegalStateException if failed to convert parameter into {@link
IonValue}.
    */
   public static QldbRevision getDocumentById(String tableName, String
documentId) {
       try {
           final IonValue ionValue =
Constants.MAPPER.writeValueAsIonValue(documentId);
           Result result = ConnectToLedger.getDriver().execute(txn -> {
```

```
                return txn.execute("SELECT c.* FROM _ql_committed_" + tableName
+ " AS c BY docId "
                                   + "WHERE docId = ?", ionValue);
            });
            if (result.isEmpty()) {
                throw new IllegalStateException("Unable to retrieve document by
id " + documentId + " in table " + tableName);
            }
            return Constants.MAPPER.readValue(result.iterator().next(),
QldbRevision.class);
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    /**
     * Return a list of modified document IDs as strings from a DML {@link
Result}.
     *
     * @param result
     *              The result set from a DML operation.
     * @return the list of document IDs modified by the operation.
     */
    public static List<String> getDocumentIdsFromDmlResult(final Result result)
{
        final List<String> strings = new ArrayList<>();
        result.iterator().forEachRemaining(row ->
strings.add(getDocumentIdFromDmlResultDocument(row)));
        return strings;
    }

    /**
     * Convert the given DML result row's document ID to string.
     *
     * @param dmlResultDocument
     *              The {@link IonValue} representing the results of a DML
operation.
     * @return a string of document ID.
     */
    public static String getDocumentIdFromDmlResultDocument(final IonValue
dmlResultDocument) {
        try {
            DmlResultDocument result =
Constants.MAPPER.readValue(dmlResultDocument, DmlResultDocument.class);
```

```
            return result.getDocumentId();
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    /**
     * Get the String value of a given {@link IonStruct} field name.
     * @param struct the {@link IonStruct} from which to get the value.
     * @param fieldName the name of the field from which to get the value.
     * @return the String value of the field within the given {@link IonStruct}.
     */
    public static String getStringValueOfStructField(final IonStruct struct,
final String fieldName) {
        return ((IonString) struct.get(fieldName)).stringValue();
    }

    /**
     * Return a copy of the given driver's license with updated person Id.
     *
     * @param oldLicense
     *              The old driver's license to update.
     * @param personId
     *              The PersonId of the driver.
     * @return the updated {@link DriversLicense}.
     */
    public static DriversLicense updatePersonIdDriversLicense(final
DriversLicense oldLicense, final String personId) {
        return new DriversLicense(personId, oldLicense.getLicenseNumber(),
oldLicense.getLicenseType(),
                oldLicense.getValidFromDate(), oldLicense.getValidToDate());
    }

    /**
     * Return a copy of the given vehicle registration with updated person Id.
     *
     * @param oldRegistration
     *              The old vehicle registration to update.
     * @param personId
     *              The PersonId of the driver.
     * @return the updated {@link VehicleRegistration}.
     */
    public static VehicleRegistration updateOwnerVehicleRegistration(final
VehicleRegistration oldRegistration,
```

```
                                                                    final
 String personId) {
        return new VehicleRegistration(oldRegistration.getVin(),
 oldRegistration.getLicensePlateNumber(),
                oldRegistration.getState(), oldRegistration.getCity(),
 oldRegistration.getPendingPenaltyTicketAmount(),
                oldRegistration.getValidFromDate(),
 oldRegistration.getValidToDate(),
                new Owners(new Owner(personId), Collections.emptyList()));
    }
}
```

1.x

> **Important**
>
> Pour le package Amazon Ion, vous devez utiliser l'espace de nomscom.amazon.ion dans votre application. AWS SDK for JavaCela dépend d'un autre package Ion dans l'espace de nomssoftware.amazon.ion, mais il s'agit d'un package existant qui n'est pas compatible avec le pilote QLDB.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
```

```
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.model;

import com.amazon.ion.IonString;
import com.amazon.ion.IonStruct;
import com.amazon.ion.IonValue;
import software.amazon.qldb.QldbSession;
import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.Constants;
import software.amazon.qldb.tutorial.qldb.DmlResultDocument;
import software.amazon.qldb.tutorial.qldb.QldbRevision;

import java.io.IOException;

import java.math.BigDecimal;
import java.text.ParseException;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;

/**
 * Sample domain objects for use throughout this tutorial.
 */
public final class SampleData {
    public static final DateTimeFormatter DATE_TIME_FORMAT =
 DateTimeFormatter.ofPattern("yyyy-MM-dd");

    public static final List<VehicleRegistration> REGISTRATIONS =
 Collections.unmodifiableList(Arrays.asList(
            new VehicleRegistration("1N4AL11D75C109151", "LEWISR261LL", "WA",
 "Seattle",
                    BigDecimal.valueOf(90.25), convertToLocalDate("2017-08-21"),
 convertToLocalDate("2020-05-11"),
                    new Owners(new Owner(null), Collections.emptyList())),
```

```
            new VehicleRegistration("KM8SRDHF6EU074761", "CA762X", "WA", "Kent",
                    BigDecimal.valueOf(130.75),
convertToLocalDate("2017-09-14"), convertToLocalDate("2020-06-25"),
                    new Owners(new Owner(null), Collections.emptyList())),
            new VehicleRegistration("3HGGK5G53FM761765", "CD820Z", "WA",
"Everett",
                    BigDecimal.valueOf(442.30),
convertToLocalDate("2011-03-17"), convertToLocalDate("2021-03-24"),
                    new Owners(new Owner(null), Collections.emptyList())),
            new VehicleRegistration("1HVBBAANXWH544237", "LS477D", "WA",
"Tacoma",
                    BigDecimal.valueOf(42.20), convertToLocalDate("2011-10-26"),
convertToLocalDate("2023-09-25"),
                    new Owners(new Owner(null), Collections.emptyList())),
            new VehicleRegistration("1C4RJFAG0FC625797", "TH393F", "WA",
"Olympia",
                    BigDecimal.valueOf(30.45), convertToLocalDate("2013-09-02"),
convertToLocalDate("2024-03-19"),
                    new Owners(new Owner(null), Collections.emptyList()))
    ));

    public static final List<Vehicle> VEHICLES =
Collections.unmodifiableList(Arrays.asList(
            new Vehicle("1N4AL11D75C109151", "Sedan", 2011,  "Audi", "A5",
"Silver"),
            new Vehicle("KM8SRDHF6EU074761", "Sedan", 2015, "Tesla", "Model S",
"Blue"),
            new Vehicle("3HGGK5G53FM761765", "Motorcycle", 2011, "Ducati",
"Monster 1200", "Yellow"),
            new Vehicle("1HVBBAANXWH544237", "Semi", 2009, "Ford", "F 150",
"Black"),
            new Vehicle("1C4RJFAG0FC625797", "Sedan", 2019, "Mercedes", "CLK
350", "White")
    ));

    public static final List<Person> PEOPLE =
Collections.unmodifiableList(Arrays.asList(
            new Person("Raul", "Lewis", convertToLocalDate("1963-08-19"),
                    "LEWISR261LL", "Driver License",  "1719 University Street,
Seattle, WA, 98109"),
            new Person("Brent", "Logan", convertToLocalDate("1967-07-03"),
                    "LOGANB486CG", "Driver License", "43 Stockert Hollow Road,
Everett, WA, 98203"),
            new Person("Alexis", "Pena", convertToLocalDate("1974-02-10"),
```

```
                    "744 849 301", "SSN", "4058 Melrose Street, Spokane Valley,
WA, 99206"),
            new Person("Melvin", "Parker", convertToLocalDate("1976-05-22"),
                    "P626-168-229-765", "Passport", "4362 Ryder Avenue, Seattle,
WA, 98101"),
            new Person("Salvatore", "Spencer", convertToLocalDate("1997-11-15"),
                    "S152-780-97-415-0", "Passport", "4450 Honeysuckle Lane,
Seattle, WA, 98101")
    ));

    public static final List<DriversLicense> LICENSES =
Collections.unmodifiableList(Arrays.asList(
            new DriversLicense(null, "LEWISR261LL", "Learner",
                    convertToLocalDate("2016-12-20"),
convertToLocalDate("2020-11-15")),
            new DriversLicense(null, "LOGANB486CG", "Probationary",
                    convertToLocalDate("2016-04-06"),
convertToLocalDate("2020-11-15")),
            new DriversLicense(null, "744 849 301", "Full",
                    convertToLocalDate("2017-12-06"),
convertToLocalDate("2022-10-15")),
            new DriversLicense(null, "P626-168-229-765", "Learner",
                    convertToLocalDate("2017-08-16"),
convertToLocalDate("2021-11-15")),
            new DriversLicense(null, "S152-780-97-415-0", "Probationary",
                    convertToLocalDate("2015-08-15"),
convertToLocalDate("2021-08-21"))
    ));

    private SampleData() { }

    /**
     * Converts a date string with the format 'yyyy-MM-dd' into a {@link
java.util.Date} object.
     *
     * @param date
     *              The date string to convert.
     * @return {@link LocalDate} or null if there is a {@link ParseException}
     */
    public static synchronized LocalDate convertToLocalDate(String date) {
        return LocalDate.parse(date, DATE_TIME_FORMAT);
    }

    /**
```

```
     * Convert the result set into a list of IonValues.
     *
     * @param result
     *              The result set to convert.
     * @return a list of IonValues.
     */
    public static List<IonValue> toIonValues(Result result) {
        final List<IonValue> valueList = new ArrayList<>();
        result.iterator().forEachRemaining(valueList::add);
        return valueList;
    }

    /**
     * Get the document ID of a particular document.
     *
     * @param txn
     *              A transaction executor object.
     * @param tableName
     *              Name of the table containing the document.
     * @param identifier
     *              The identifier used to narrow down the search.
     * @param value
     *              Value of the identifier.
     * @return the list of document IDs in the result set.
     */
    public static String getDocumentId(final TransactionExecutor txn, final
String tableName,
                                       final String identifier, final String
value) {
        try {
            final List<IonValue> parameters =
Collections.singletonList(Constants.MAPPER.writeValueAsIonValue(value));
            final String query = String.format("SELECT metadata.id FROM
_ql_committed_%s AS p WHERE p.data.%s = ?",
                    tableName, identifier);
            Result result = txn.execute(query, parameters);
            if (result.isEmpty()) {
                throw new IllegalStateException("Unable to retrieve document ID
using " + value);
            }
            return getStringValueOfStructField((IonStruct)
result.iterator().next(), "id");
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
```

```
        }
    }

    /**
     * Get the document by ID.
     *
     * @param qldbSession
     *              A QLDB session.
     * @param tableName
     *              Name of the table to insert documents into.
     * @param documentId
     *              The unique ID of a document in the Person table.
     * @return a {@link QldbRevision} object.
     * @throws IllegalStateException if failed to convert parameter into {@link
IonValue}.
     */
    public static QldbRevision getDocumentById(QldbSession qldbSession, String
tableName, String documentId) {
        try {
            final List<IonValue> parameters =
Collections.singletonList(Constants.MAPPER.writeValueAsIonValue(documentId));
            final String query = String.format("SELECT c.* FROM _ql_committed_%s
AS c BY docId WHERE docId = ?", tableName);
            Result result = qldbSession.execute(query, parameters);
            if (result.isEmpty()) {
                throw new IllegalStateException("Unable to retrieve document by
id " + documentId + " in table " + tableName);
            }
            return Constants.MAPPER.readValue(result.iterator().next(),
QldbRevision.class);
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    /**
     * Return a list of modified document IDs as strings from a DML {@link
Result}.
     *
     * @param result
     *              The result set from a DML operation.
     * @return the list of document IDs modified by the operation.
     */
```

```
   public static List<String> getDocumentIdsFromDmlResult(final Result result)
{
       final List<String> strings = new ArrayList<>();
       result.iterator().forEachRemaining(row ->
strings.add(getDocumentIdFromDmlResultDocument(row)));
       return strings;
   }

   /**
    * Convert the given DML result row's document ID to string.
    *
    * @param dmlResultDocument
    *              The {@link IonValue} representing the results of a DML
operation.
    * @return a string of document ID.
    */
   public static String getDocumentIdFromDmlResultDocument(final IonValue
dmlResultDocument) {
       try {
           DmlResultDocument result =
Constants.MAPPER.readValue(dmlResultDocument, DmlResultDocument.class);
           return result.getDocumentId();
       } catch (IOException ioe) {
           throw new IllegalStateException(ioe);
       }
   }

   /**
    * Get the String value of a given {@link IonStruct} field name.
    * @param struct the {@link IonStruct} from which to get the value.
    * @param fieldName the name of the field from which to get the value.
    * @return the String value of the field within the given {@link IonStruct}.
    */
   public static String getStringValueOfStructField(final IonStruct struct,
final String fieldName) {
       return ((IonString) struct.get(fieldName)).stringValue();
   }

   /**
    * Return a copy of the given driver's license with updated person Id.
    *
    * @param oldLicense
    *              The old driver's license to update.
    * @param personId
```

```
     *              The PersonId of the driver.
     * @return the updated {@link DriversLicense}.
     */
    public static DriversLicense updatePersonIdDriversLicense(final
 DriversLicense oldLicense, final String personId) {
        return new DriversLicense(personId, oldLicense.getLicenseNumber(),
 oldLicense.getLicenseType(),
                oldLicense.getValidFromDate(), oldLicense.getValidToDate());
    }

    /**
     * Return a copy of the given vehicle registration with updated person Id.
     *
     * @param oldRegistration
     *              The old vehicle registration to update.
     * @param personId
     *              The PersonId of the driver.
     * @return the updated {@link VehicleRegistration}.
     */
    public static VehicleRegistration updateOwnerVehicleRegistration(final
 VehicleRegistration oldRegistration,
                                                                     final
 String personId) {
        return new VehicleRegistration(oldRegistration.getVin(),
 oldRegistration.getLicensePlateNumber(),
                oldRegistration.getState(), oldRegistration.getCity(),
 oldRegistration.getPendingPenaltyTicketAmount(),
                oldRegistration.getValidFromDate(),
 oldRegistration.getValidToDate(),
                new Owners(new Owner(personId), Collections.emptyList()));
    }
}
```

Note

- Cette classe utilise des bibliothèques Ion pour fournir des méthodes d'assistance qui convertissent vos données vers et depuis le format Ion.
- `LagetDocumentId` méthode exécute une requête sur une table avec le préfixe`_ql_committed_`. Il s'agit d'un préfixe réservé qui signifie que vous souhaitez

interroger la vue validée d'une table. Dans cette vue, vos données sont imbriquées dans le`data` champ et les métadonnées sont imbriquées dans le`metadata` champ.

3. Compilez et exécutez le programme suivant (`CreateTable.java`) pour créer les tables mentionnées précédemment.

   2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.SampleData;
```

```
/**
 * Create tables in a QLDB ledger.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class CreateTable {
    public static final Logger log = LoggerFactory.getLogger(CreateTable.class);

    private CreateTable() { }

    /**
     * Registrations, vehicles, owners, and licenses tables being created in a
 single transaction.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param tableName
     *              Name of the table to be created.
     * @return the number of tables created.
     */
    public static int createTable(final TransactionExecutor txn, final String
 tableName) {
        log.info("Creating the '{}' table...", tableName);
        final String createTable = String.format("CREATE TABLE %s", tableName);
        final Result result = txn.execute(createTable);
        log.info("{} table created successfully.", tableName);
        return SampleData.toIonValues(result).size();
    }

    public static void main(final String... args) {
        ConnectToLedger.getDriver().execute(txn -> {
            createTable(txn, Constants.DRIVERS_LICENSE_TABLE_NAME);
            createTable(txn, Constants.PERSON_TABLE_NAME);
            createTable(txn, Constants.VEHICLE_TABLE_NAME);
            createTable(txn, Constants.VEHICLE_REGISTRATION_TABLE_NAME);
        });
    }
}
```

1.x

```
/*
 * Copyright 2020 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.SampleData;

/**
 * Create tables in a QLDB ledger.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
```

```
 */
public final class CreateTable {
    public static final Logger log = LoggerFactory.getLogger(CreateTable.class);

    private CreateTable() { }

    /**
     * Registrations, vehicles, owners, and licenses tables being created in a
 single transaction.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param tableName
     *              Name of the table to be created.
     * @return the number of tables created.
     */
    public static int createTable(final TransactionExecutor txn, final String
 tableName) {
        log.info("Creating the '{}' table...", tableName);
        final String createTable = String.format("CREATE TABLE %s", tableName);
        final Result result = txn.execute(createTable);
        log.info("{} table created successfully.", tableName);
        return SampleData.toIonValues(result).size();
    }

    public static void main(final String... args) {
        ConnectToLedger.getDriver().execute(txn -> {
            createTable(txn, Constants.DRIVERS_LICENSE_TABLE_NAME);
            createTable(txn, Constants.PERSON_TABLE_NAME);
            createTable(txn, Constants.VEHICLE_TABLE_NAME);
            createTable(txn, Constants.VEHICLE_REGISTRATION_TABLE_NAME);
        }, (retryAttempt) -> log.info("Retrying due to OCC conflict..."));
    }
}
```

> **Note**
>
> Ce programme montre comment transmettre un`TransactionExecutor` lambda à la`execute` méthode. Dans cet exemple, vous exécutez plusieurs instructions`CREATE TABLE` PartiQL dans une seule transaction à l'aide d'une expression lambda.

La `execute` méthode démarre implicitement une transaction, exécute toutes les instructions du lambda, puis valide automatiquement la transaction.

4. Compilez et exécutez le programme suivant (`CreateIndex.java`) pour créer des index sur les tables, comme décrit précédemment.

   2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;


import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.SampleData;
```

```
/**
 * Create indexes on tables in a particular ledger.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class CreateIndex {
    public static final Logger log = LoggerFactory.getLogger(CreateIndex.class);

    private CreateIndex() { }

    /**
     * In this example, create indexes for registrations and vehicles tables.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param tableName
     *              Name of the table to be created.
     * @param indexAttribute
     *              The index attribute to use.
     * @return the number of tables created.
     */
    public static int createIndex(final TransactionExecutor txn, final String
 tableName, final String indexAttribute) {
        log.info("Creating an index on {}...", indexAttribute);
        final String createIndex = String.format("CREATE INDEX ON %s (%s)",
 tableName, indexAttribute);
        final Result r = txn.execute(createIndex);
        return SampleData.toIonValues(r).size();
    }

    public static void main(final String... args) {
        ConnectToLedger.getDriver().execute(txn -> {
            createIndex(txn, Constants.PERSON_TABLE_NAME,
 Constants.PERSON_GOV_ID_INDEX_NAME);
            createIndex(txn, Constants.VEHICLE_TABLE_NAME,
 Constants.VIN_INDEX_NAME);
            createIndex(txn, Constants.DRIVERS_LICENSE_TABLE_NAME,
 Constants.DRIVER_LICENSE_NUMBER_INDEX_NAME);
            createIndex(txn, Constants.DRIVERS_LICENSE_TABLE_NAME,
 Constants.DRIVER_LICENSE_PERSONID_INDEX_NAME);
```

```
            createIndex(txn, Constants.VEHICLE_REGISTRATION_TABLE_NAME,
 Constants.VIN_INDEX_NAME);
            createIndex(txn, Constants.VEHICLE_REGISTRATION_TABLE_NAME,

 Constants.VEHICLE_REGISTRATION_LICENSE_PLATE_NUMBER_INDEX_NAME);
        });
        log.info("Indexes created successfully!");
    }
}
```

1.x

```
/*
 * Copyright 2020 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

```
import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.SampleData;

/**
 * Create indexes on tables in a particular ledger.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class CreateIndex {
    public static final Logger log = LoggerFactory.getLogger(CreateIndex.class);

    private CreateIndex() { }

    /**
     * In this example, create indexes for registrations and vehicles tables.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param tableName
     *              Name of the table to be created.
     * @param indexAttribute
     *              The index attribute to use.
     * @return the number of tables created.
     */
    public static int createIndex(final TransactionExecutor txn, final String
 tableName, final String indexAttribute) {
        log.info("Creating an index on {}...", indexAttribute);
        final String createIndex = String.format("CREATE INDEX ON %s (%s)",
 tableName, indexAttribute);
        final Result r = txn.execute(createIndex);
        return SampleData.toIonValues(r).size();
    }

    public static void main(final String... args) {
        ConnectToLedger.getDriver().execute(txn -> {
            createIndex(txn, Constants.PERSON_TABLE_NAME,
 Constants.PERSON_GOV_ID_INDEX_NAME);
            createIndex(txn, Constants.VEHICLE_TABLE_NAME,
 Constants.VIN_INDEX_NAME);
            createIndex(txn, Constants.DRIVERS_LICENSE_TABLE_NAME,
 Constants.DRIVER_LICENSE_NUMBER_INDEX_NAME);
```

```
            createIndex(txn, Constants.DRIVERS_LICENSE_TABLE_NAME,
 Constants.DRIVER_LICENSE_PERSONID_INDEX_NAME);
            createIndex(txn, Constants.VEHICLE_REGISTRATION_TABLE_NAME,
 Constants.VIN_INDEX_NAME);
            createIndex(txn, Constants.VEHICLE_REGISTRATION_TABLE_NAME,

 Constants.VEHICLE_REGISTRATION_LICENSE_PLATE_NUMBER_INDEX_NAME);
        }, (retryAttempt) -> log.info("Retrying due to OCC conflict..."));
        log.info("Indexes created successfully!");
    }
}
```

5. Compilez et exécutez le programme suivant (`InsertDocument.java`) pour insérer les exemples de données dans vos tables.

   2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */
```

```
package software.amazon.qldb.tutorial;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.amazon.ion.IonValue;

import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.DriversLicense;
import software.amazon.qldb.tutorial.model.SampleData;
import software.amazon.qldb.tutorial.model.VehicleRegistration;

/**
 * Insert documents into a table in a QLDB ledger.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class InsertDocument {
    public static final Logger log =
 LoggerFactory.getLogger(InsertDocument.class);

    private InsertDocument() { }

    /**
     * Insert the given list of documents into the specified table and return
 the document IDs of the inserted documents.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param tableName
     *              Name of the table to insert documents into.
     * @param documents
     *              List of documents to insert into the specified table.
     * @return a list of document IDs.
     * @throws IllegalStateException if failed to convert documents into an
 {@link IonValue}.
     */
```

```
   public static List<String> insertDocuments(final TransactionExecutor txn,
final String tableName,
                                              final List documents) {
       log.info("Inserting some documents in the {} table...", tableName);
       try {
           final String query = String.format("INSERT INTO %s ?", tableName);
           final IonValue ionDocuments =
Constants.MAPPER.writeValueAsIonValue(documents);

           return SampleData.getDocumentIdsFromDmlResult(txn.execute(query,
ionDocuments));
       } catch (IOException ioe) {
           throw new IllegalStateException(ioe);
       }
   }

   /**
    * Update PersonIds in driver's licenses and in vehicle registrations using
document IDs.
    *
    * @param documentIds
    *              List of document IDs representing the PersonIds in
DriversLicense and PrimaryOwners in VehicleRegistration.
    * @param licenses
    *              List of driver's licenses to update.
    * @param registrations
    *              List of registrations to update.
    */
   public static void updatePersonId(final List<String> documentIds, final
List<DriversLicense> licenses,
                                     final List<VehicleRegistration>
registrations) {
       for (int i = 0; i < documentIds.size(); ++i) {
           DriversLicense license = SampleData.LICENSES.get(i);
           VehicleRegistration registration = SampleData.REGISTRATIONS.get(i);
           licenses.add(SampleData.updatePersonIdDriversLicense(license,
documentIds.get(i)));

registrations.add(SampleData.updateOwnerVehicleRegistration(registration,
documentIds.get(i)));
       }
   }

   public static void main(final String... args) {
```

```
        final List<DriversLicense> newDriversLicenses = new ArrayList<>();
        final List<VehicleRegistration> newVehicleRegistrations = new
 ArrayList<>();
        ConnectToLedger.getDriver().execute(txn -> {
            List<String> documentIds = insertDocuments(txn,
 Constants.PERSON_TABLE_NAME, SampleData.PEOPLE);
            updatePersonId(documentIds, newDriversLicenses,
 newVehicleRegistrations);
            insertDocuments(txn, Constants.VEHICLE_TABLE_NAME,
 SampleData.VEHICLES);
            insertDocuments(txn, Constants.VEHICLE_REGISTRATION_TABLE_NAME,
                    Collections.unmodifiableList(newVehicleRegistrations));
            insertDocuments(txn, Constants.DRIVERS_LICENSE_TABLE_NAME,
                    Collections.unmodifiableList(newDriversLicenses));
        });
        log.info("Documents inserted successfully!");
    }
}
```

1.x

```
/*
 * Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
```

```
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazon.ion.IonValue;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.qldb.QldbSession;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.DriversLicense;
import software.amazon.qldb.tutorial.model.SampleData;
import software.amazon.qldb.tutorial.model.VehicleRegistration;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

/**
 * Insert documents into a table in a QLDB ledger.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class InsertDocument {
    public static final Logger log =
 LoggerFactory.getLogger(InsertDocument.class);

    private InsertDocument() { }

    /**
     * Insert the given list of documents into the specified table and return
 the document IDs of the inserted documents.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param tableName
     *              Name of the table to insert documents into.
     * @param documents
     *              List of documents to insert into the specified table.
```

```
    * @return a list of document IDs.
    * @throws IllegalStateException if failed to convert documents into an
{@link IonValue}.
    */
   public static List<String> insertDocuments(final TransactionExecutor txn,
final String tableName,
                                              final List documents) {
       log.info("Inserting some documents in the {} table...", tableName);
       try {
           final String statement = String.format("INSERT INTO %s ?",
tableName);
           final IonValue ionDocuments =
Constants.MAPPER.writeValueAsIonValue(documents);
           final List<IonValue> parameters =
Collections.singletonList(ionDocuments);
           return SampleData.getDocumentIdsFromDmlResult(txn.execute(statement,
parameters));
       } catch (IOException ioe) {
           throw new IllegalStateException(ioe);
       }
   }

   /**
    * Update PersonIds in driver's licenses and in vehicle registrations using
document IDs.
    *
    * @param documentIds
    *              List of document IDs representing the PersonIds in
DriversLicense and PrimaryOwners in VehicleRegistration.
    * @param licenses
    *              List of driver's licenses to update.
    * @param registrations
    *              List of registrations to update.
    */
   public static void updatePersonId(final List<String> documentIds, final
List<DriversLicense> licenses,
                                     final List<VehicleRegistration>
registrations) {
       for (int i = 0; i < documentIds.size(); ++i) {
           DriversLicense license = SampleData.LICENSES.get(i);
           VehicleRegistration registration = SampleData.REGISTRATIONS.get(i);
           licenses.add(SampleData.updatePersonIdDriversLicense(license,
documentIds.get(i)));
```

```
        registrations.add(SampleData.updateOwnerVehicleRegistration(registration,
        documentIds.get(i)));
            }
        }

        public static void main(final String... args) {
            final List<DriversLicense> newDriversLicenses = new ArrayList<>();
            final List<VehicleRegistration> newVehicleRegistrations = new
        ArrayList<>();
            ConnectToLedger.getDriver().execute(txn -> {
                List<String> documentIds = insertDocuments(txn,
        Constants.PERSON_TABLE_NAME, SampleData.PEOPLE);
                updatePersonId(documentIds, newDriversLicenses,
        newVehicleRegistrations);
                insertDocuments(txn, Constants.VEHICLE_TABLE_NAME,
        SampleData.VEHICLES);
                insertDocuments(txn, Constants.VEHICLE_REGISTRATION_TABLE_NAME,
                        Collections.unmodifiableList(newVehicleRegistrations));
                insertDocuments(txn, Constants.DRIVERS_LICENSE_TABLE_NAME,
                        Collections.unmodifiableList(newDriversLicenses));
            }, (retryAttempt) -> log.info("Retrying due to OCC conflict..."));
            log.info("Documents inserted successfully!");
        }
    }
```

> **Note**
>
> - Ce programme montre comment appeler la`execute` méthode avec des valeurs paramétrées. Vous pouvez transmettre des paramètres de données de type`IonValue` en plus de l'instruction PartiQL que vous souhaitez exécuter. Utilisez un point d'interrogation (?) comme espace réservé à une variable dans votre chaîne de déclaration.
> - Si une`INSERT` instruction aboutit, elle renvoie la valeur`id` de chaque document inséré.

Vous pouvez ensuite utiliser`SELECT` des instructions pour lire les données des tables du`vehicle-registration` registre. Passez à Étape 4 : interroger les tables d'un registre.

Étape 4 : interroger les tables d'un registre

Après avoir créé des tables dans un registre Amazon QLDB et les avoir chargées avec des données, vous pouvez exécuter des requêtes pour vérifier les données d'immatriculation du véhicule que vous venez d'insérer. QLDB utilise PartiQL comme langage de requête et Amazon Ion comme modèle de données orienté document.

PartiQL est un langage de requête open source compatible SQL qui a été étendu pour fonctionner avec Ion. Avec PartiQL, vous pouvez insérer, interroger et gérer vos données à l'aide d'opérateurs SQL familiers. Amazon Ion est un sur-ensemble de JSON. Ion est un format de données open source basé sur des documents qui vous permet de stocker et de traiter des données structurées, semi-structurées et imbriquées.

Au cours de cette étape, vous utilisez des`SELECT` instructions pour lire les données des tables du`vehicle-registration` registre.

> **Warning**
>
> Lorsque vous exécutez une requête dans QLDB sans recherche indexée, elle appelle une analyse complète de la table. PartiQL prend en charge de telles requêtes car il est compatible avec SQL. Toutefois, n'exécutez pas d'analyses de tables pour des cas d'utilisation en production dans QLDB. Les analyses de tables peuvent entraîner des problèmes de performances sur des tables de grande taille, notamment des conflits de simultanéité et des délais de transaction.
>
> Pour éviter de scanner des tables, vous devez exécuter des instructions avec une clause de`WHERE` prédicat à l'aide d'un opérateur d'égalité sur un champ indexé ou un identifiant de document, par exemple,`WHERE indexedField = 123` ou`WHERE indexedField IN (456, 789)`. Pour plus d'informations, veuillez consulter Optimisation des performances des données.

Pour interroger les tables

- Compilez et exécutez le programme suivant (`FindVehicles.java`) pour interroger tous les véhicules enregistrés par une personne dans votre registre.

  2.x

```
/*
```

```
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import java.io.IOException;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.amazon.ion.IonValue;

import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.Person;
import software.amazon.qldb.tutorial.model.SampleData;

/**
 * Find all vehicles registered under a person.
 *
 * This code expects that you have AWS credentials setup per:
```

```
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class FindVehicles {
    public static final Logger log =
 LoggerFactory.getLogger(FindVehicles.class);

    private FindVehicles() { }

    /**
     * Find vehicles registered under a driver using their government ID.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param govId
     *              The government ID of the owner.
     * @throws IllegalStateException if failed to convert parameters into {@link
 IonValue}.
     */
    public static void findVehiclesForOwner(final TransactionExecutor txn, final
 String govId) {
        try {
            final String documentId = Person.getDocumentIdByGovId(txn, govId);
            final String query = "SELECT v FROM Vehicle AS v INNER JOIN
 VehicleRegistration AS r "
                    + "ON v.VIN = r.VIN WHERE r.Owners.PrimaryOwner.PersonId
 = ?";

            final Result result = txn.execute(query,
 Constants.MAPPER.writeValueAsIonValue(documentId));
            log.info("List of Vehicles for owner with GovId: {}...", govId);
            ScanTable.printDocuments(result);
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    public static void main(final String... args) {
        final Person person = SampleData.PEOPLE.get(0);
        ConnectToLedger.getDriver().execute(txn -> {
            findVehiclesForOwner(txn, person.getGovId());
        });
    }
```

```
}
```

1.x

```
/*
 * Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import java.io.IOException;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.amazon.ion.IonValue;

import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.Person;
import software.amazon.qldb.tutorial.model.SampleData;
```

```
/**
 * Find all vehicles registered under a person.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class FindVehicles {
    public static final Logger log =
 LoggerFactory.getLogger(FindVehicles.class);

    private FindVehicles() { }

    /**
     * Find vehicles registered under a driver using their government ID.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param govId
     *              The government ID of the owner.
     * @throws IllegalStateException if failed to convert parameters into {@link
 IonValue}.
     */
    public static void findVehiclesForOwner(final TransactionExecutor txn, final
 String govId) {
        try {
            final String documentId = Person.getDocumentIdByGovId(txn, govId);
            final String query = "SELECT v FROM Vehicle AS v INNER JOIN
 VehicleRegistration AS r "
                    + "ON v.VIN = r.VIN WHERE r.Owners.PrimaryOwner.PersonId
 = ?";

            final Result result = txn.execute(query,
 Constants.MAPPER.writeValueAsIonValue(documentId));
            log.info("List of Vehicles for owner with GovId: {}...", govId);
            ScanTable.printDocuments(result);
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    public static void main(final String... args) {
        final Person person = SampleData.PEOPLE.get(0);
```

```
        ConnectToLedger.getDriver().execute(txn -> {
            findVehiclesForOwner(txn, person.getGovId());
        }, (retryAttempt) -> log.info("Retrying due to OCC conflict..."));
    }
}
```

> **Note**
>
> Tout d'abord, ce programme interroge la`Person` table contenant le document`GovId` `LEWISR261LL` pour obtenir son champ `id` métadonnées.
> Il utilise ensuite ce document`id` comme clé étrangère pour interroger la`VehicleRegistration` table`PrimaryOwner.PersonId`. Il se joint également`VehicleRegistration` à la`Vehicle` table sur le`VIN` terrain.

Pour en savoir plus sur la modification de documents dans les tableaux du`vehicle-registration` grand livre, consultezÉtape 5 : Modifier des documents dans un registre.

Étape 5 : Modifier des documents dans un registre

Maintenant que vous avez des données sur lesquelles travailler, vous pouvez commencer à apporter des modifications aux documents du`vehicle-registration` registre dans Amazon QLDB. Dans cette étape, les exemples de code suivants montrent comment exécuter des instructions de langage de données (DML). Ces déclarations mettent à jour le nom du propriétaire principal d'un véhicule et ajoutent un propriétaire secondaire à un autre véhicule.

Pour modifier des documents

1. Compilez et exécutez le programme suivant (`TransferVehicleOwnership.java`) pour mettre à jour le nom du propriétaire principal du véhicule avec le `VIN1N4AL11D75C109151` dans votre registre.

   2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
```

```
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazon.ion.IonReader;
import com.amazon.ion.IonStruct;
import com.amazon.ion.IonValue;
import com.amazon.ion.system.IonReaderBuilder;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.LinkedHashMap;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.Owner;
import software.amazon.qldb.tutorial.model.Person;
import software.amazon.qldb.tutorial.model.SampleData;
```

```
/**
 * Find primary owner for a particular vehicle's VIN.
 * Transfer to another primary owner for a particular vehicle's VIN.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class TransferVehicleOwnership {
    public static final Logger log =
 LoggerFactory.getLogger(TransferVehicleOwnership.class);

    private TransferVehicleOwnership() { }

    /**
     * Query a driver's information using the given ID.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param documentId
     *              The unique ID of a document in the Person table.
     * @return a {@link Person} object.
     * @throws IllegalStateException if failed to convert parameter into {@link
 IonValue}.
     */
    public static Person findPersonFromDocumentId(final TransactionExecutor txn,
 final String documentId) {
        try {
            log.info("Finding person for documentId: {}...", documentId);
            final String query = "SELECT p.* FROM Person AS p BY pid WHERE pid
 = ?";

            Result result = txn.execute(query,
 Constants.MAPPER.writeValueAsIonValue(documentId));
            if (result.isEmpty()) {
                throw new IllegalStateException("Unable to find person with ID:
 " + documentId);
            }

            return Constants.MAPPER.readValue(result.iterator().next(),
 Person.class);
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
```

```
    }

    /**
     * Find the primary owner for the given VIN.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param vin
     *              Unique VIN for a vehicle.
     * @return a {@link Person} object.
     * @throws IllegalStateException if failed to convert parameter into {@link
IonValue}.
     */
    public static Person findPrimaryOwnerForVehicle(final TransactionExecutor
txn, final String vin) {
        try {
            log.info("Finding primary owner for vehicle with Vin: {}...", vin);
            final String query = "SELECT Owners.PrimaryOwner.PersonId FROM
VehicleRegistration AS v WHERE v.VIN = ?";
            final List<IonValue> parameters =
Collections.singletonList(Constants.MAPPER.writeValueAsIonValue(vin));
            Result result = txn.execute(query, parameters);
            final List<IonStruct> documents = ScanTable.toIonStructs(result);
            ScanTable.printDocuments(documents);
            if (documents.isEmpty()) {
                throw new IllegalStateException("Unable to find registrations
with VIN: " + vin);
            }

            final IonReader reader =
IonReaderBuilder.standard().build(documents.get(0));
            final String personId = Constants.MAPPER.readValue(reader,
LinkedHashMap.class).get("PersonId").toString();
            return findPersonFromDocumentId(txn, personId);
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    /**
     * Update the primary owner for a vehicle registration with the given
documentId.
     *
     * @param txn
```

```
     *              The {@link TransactionExecutor} for lambda execute.
     * @param vin
     *              Unique VIN for a vehicle.
     * @param documentId
     *              New PersonId for the primary owner.
     * @throws IllegalStateException if no vehicle registration was found using
the given document ID and VIN, or if failed
     * to convert parameters into {@link IonValue}.
     */
    public static void updateVehicleRegistration(final TransactionExecutor txn,
final String vin, final String documentId) {
        try {
            log.info("Updating primary owner for vehicle with Vin: {}...", vin);
            final String query = "UPDATE VehicleRegistration AS v SET
v.Owners.PrimaryOwner = ? WHERE v.VIN = ?";

            final List<IonValue> parameters = new ArrayList<>();
            parameters.add(Constants.MAPPER.writeValueAsIonValue(new
Owner(documentId)));
            parameters.add(Constants.MAPPER.writeValueAsIonValue(vin));

            Result result = txn.execute(query, parameters);
            ScanTable.printDocuments(result);
            if (result.isEmpty()) {
                throw new IllegalStateException("Unable to transfer vehicle,
could not find registration.");
            } else {
                log.info("Successfully transferred vehicle with VIN '{}' to new
owner.", vin);
            }
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    public static void main(final String... args) {
        final String vin = SampleData.VEHICLES.get(0).getVin();
        final String primaryOwnerGovId = SampleData.PEOPLE.get(0).getGovId();
        final String newPrimaryOwnerGovId = SampleData.PEOPLE.get(1).getGovId();

        ConnectToLedger.getDriver().execute(txn -> {
            final Person primaryOwner = findPrimaryOwnerForVehicle(txn, vin);
            if (!primaryOwner.getGovId().equals(primaryOwnerGovId)) {
                // Verify the primary owner.
```

```
                throw new IllegalStateException("Incorrect primary owner
 identified for vehicle, unable to transfer.");
            }

            final String newOwner = Person.getDocumentIdByGovId(txn,
 newPrimaryOwnerGovId);
            updateVehicleRegistration(txn, vin, newOwner);
        });
        log.info("Successfully transferred vehicle ownership!");
    }
}
```

1.x

```
/*
 * Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazon.ion.IonReader;
```

```
import com.amazon.ion.IonStruct;
import com.amazon.ion.IonValue;
import com.amazon.ion.system.IonReaderBuilder;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.LinkedHashMap;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.Owner;
import software.amazon.qldb.tutorial.model.Person;
import software.amazon.qldb.tutorial.model.SampleData;

/**
 * Find primary owner for a particular vehicle's VIN.
 * Transfer to another primary owner for a particular vehicle's VIN.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class TransferVehicleOwnership {
    public static final Logger log =
 LoggerFactory.getLogger(TransferVehicleOwnership.class);

    private TransferVehicleOwnership() { }

    /**
     * Query a driver's information using the given ID.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param documentId
     *              The unique ID of a document in the Person table.
     * @return a {@link Person} object.
     * @throws IllegalStateException if failed to convert parameter into {@link
 IonValue}.
     */
```

```
   public static Person findPersonFromDocumentId(final TransactionExecutor txn,
final String documentId) {
       try {
           log.info("Finding person for documentId: {}...", documentId);
           final String query = "SELECT p.* FROM Person AS p BY pid WHERE pid
= ?";

           Result result = txn.execute(query,
Constants.MAPPER.writeValueAsIonValue(documentId));
           if (result.isEmpty()) {
               throw new IllegalStateException("Unable to find person with ID:
" + documentId);
           }

           return Constants.MAPPER.readValue(result.iterator().next(),
Person.class);
       } catch (IOException ioe) {
           throw new IllegalStateException(ioe);
       }
   }

   /**
    * Find the primary owner for the given VIN.
    *
    * @param txn
    *              The {@link TransactionExecutor} for lambda execute.
    * @param vin
    *              Unique VIN for a vehicle.
    * @return a {@link Person} object.
    * @throws IllegalStateException if failed to convert parameter into {@link
IonValue}.
    */
   public static Person findPrimaryOwnerForVehicle(final TransactionExecutor
txn, final String vin) {
       try {
           log.info("Finding primary owner for vehicle with Vin: {}...", vin);
           final String query = "SELECT Owners.PrimaryOwner.PersonId FROM
VehicleRegistration AS v WHERE v.VIN = ?";
           final List<IonValue> parameters =
Collections.singletonList(Constants.MAPPER.writeValueAsIonValue(vin));
           Result result = txn.execute(query, parameters);
           final List<IonStruct> documents = ScanTable.toIonStructs(result);
           ScanTable.printDocuments(documents);
           if (documents.isEmpty()) {
```

```
                throw new IllegalStateException("Unable to find registrations
with VIN: " + vin);
            }

            final IonReader reader =
IonReaderBuilder.standard().build(documents.get(0));
            final String personId = Constants.MAPPER.readValue(reader,
LinkedHashMap.class).get("PersonId").toString();
            return findPersonFromDocumentId(txn, personId);
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    /**
     * Update the primary owner for a vehicle registration with the given
documentId.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param vin
     *              Unique VIN for a vehicle.
     * @param documentId
     *              New PersonId for the primary owner.
     * @throws IllegalStateException if no vehicle registration was found using
the given document ID and VIN, or if failed
     * to convert parameters into {@link IonValue}.
     */
    public static void updateVehicleRegistration(final TransactionExecutor txn,
final String vin, final String documentId) {
        try {
            log.info("Updating primary owner for vehicle with Vin: {}...", vin);
            final String query = "UPDATE VehicleRegistration AS v SET
v.Owners.PrimaryOwner = ? WHERE v.VIN = ?";

            final List<IonValue> parameters = new ArrayList<>();
            parameters.add(Constants.MAPPER.writeValueAsIonValue(new
Owner(documentId)));
            parameters.add(Constants.MAPPER.writeValueAsIonValue(vin));

            Result result = txn.execute(query, parameters);
            ScanTable.printDocuments(result);
            if (result.isEmpty()) {
```

```
                throw new IllegalStateException("Unable to transfer vehicle,
 could not find registration.");
            } else {
                log.info("Successfully transferred vehicle with VIN '{}' to new
 owner.", vin);
            }
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    public static void main(final String... args) {
        final String vin = SampleData.VEHICLES.get(0).getVin();
        final String primaryOwnerGovId = SampleData.PEOPLE.get(0).getGovId();
        final String newPrimaryOwnerGovId = SampleData.PEOPLE.get(1).getGovId();

        ConnectToLedger.getDriver().execute(txn -> {
            final Person primaryOwner = findPrimaryOwnerForVehicle(txn, vin);
            if (!primaryOwner.getGovId().equals(primaryOwnerGovId)) {
                // Verify the primary owner.
                throw new IllegalStateException("Incorrect primary owner
 identified for vehicle, unable to transfer.");
            }

            final String newOwner = Person.getDocumentIdByGovId(txn,
 newPrimaryOwnerGovId);
            updateVehicleRegistration(txn, vin, newOwner);
        }, (retryAttempt) -> log.info("Retrying due to OCC conflict..."));
        log.info("Successfully transferred vehicle ownership!");
    }
}
```

2. Compilez et exécutez le programme suivant (`AddSecondaryOwner.java`) pour ajouter un propriétaire secondaire au véhicule avec le VIN`KM8SRDHF6EU074761` dans votre registre.

   2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
```

```
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import java.io.IOException;
import java.util.Collections;
import java.util.Iterator;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.amazon.ion.IonValue;

import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.Owner;
import software.amazon.qldb.tutorial.model.Owners;
import software.amazon.qldb.tutorial.model.Person;
import software.amazon.qldb.tutorial.model.SampleData;

/**
 * Finds and adds secondary owners for a vehicle.
 *
 * This code expects that you have AWS credentials setup per:
```

```
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class AddSecondaryOwner {
    public static final Logger log =
 LoggerFactory.getLogger(AddSecondaryOwner.class);

    private AddSecondaryOwner() { }

    /**
     * Check whether a secondary owner has already been registered for the given
 VIN.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param vin
     *              Unique VIN for a vehicle.
     * @param secondaryOwnerId
     *              The secondary owner to add.
     * @return {@code true} if the given secondary owner has already been
 registered, {@code false} otherwise.
     * @throws IllegalStateException if failed to convert VIN to an {@link
 IonValue}.
     */
    public static boolean isSecondaryOwnerForVehicle(final TransactionExecutor
 txn, final String vin,
                                                     final String
 secondaryOwnerId) {
        try {
            log.info("Finding secondary owners for vehicle with VIN: {}...",
 vin);
            final String query = "SELECT Owners.SecondaryOwners FROM
 VehicleRegistration AS v WHERE v.VIN = ?";
            final List<IonValue> parameters =
 Collections.singletonList(Constants.MAPPER.writeValueAsIonValue(vin));
            final Result result = txn.execute(query, parameters);
            final Iterator<IonValue> itr = result.iterator();
            if (!itr.hasNext()) {
                return false;
            }

            final Owners owners = Constants.MAPPER.readValue(itr.next(),
 Owners.class);
            if (null != owners.getSecondaryOwners()) {
```

```
                for (Owner owner : owners.getSecondaryOwners()) {
                    if (secondaryOwnerId.equalsIgnoreCase(owner.getPersonId()))
{
                        return true;
                    }
                }
            }

            return false;
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    /**
     * Adds a secondary owner for the specified VIN.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param vin
     *              Unique VIN for a vehicle.
     * @param secondaryOwner
     *              The secondary owner to add.
     * @throws IllegalStateException if failed to convert parameter into an
{@link IonValue}.
     */
    public static void addSecondaryOwnerForVin(final TransactionExecutor txn,
final String vin,
                                               final String secondaryOwner) {
        try {
            log.info("Inserting secondary owner for vehicle with VIN: {}...",
vin);
            final String query = String.format("FROM VehicleRegistration AS v
WHERE v.VIN = ?" +
                    "INSERT INTO v.Owners.SecondaryOwners VALUE ?");
            final IonValue newOwner = Constants.MAPPER.writeValueAsIonValue(new
Owner(secondaryOwner));
            final IonValue vinAsIonValue =
Constants.MAPPER.writeValueAsIonValue(vin);
            Result result = txn.execute(query, vinAsIonValue, newOwner);
            log.info("VehicleRegistration Document IDs which had secondary
owners added: ");
            ScanTable.printDocuments(result);
        } catch (IOException ioe) {
```

```
            throw new IllegalStateException(ioe);
        }
    }

    public static void main(final String... args) {
        final String vin = SampleData.VEHICLES.get(1).getVin();
        final String govId = SampleData.PEOPLE.get(0).getGovId();

        ConnectToLedger.getDriver().execute(txn -> {
            final String documentId = Person.getDocumentIdByGovId(txn, govId);
            if (isSecondaryOwnerForVehicle(txn, vin, documentId)) {
                log.info("Person with ID {} has already been added as a
 secondary owner of this vehicle.", govId);
            } else {
                addSecondaryOwnerForVin(txn, vin, documentId);
            }
        });
        log.info("Secondary owners successfully updated.");
    }
}
```

1.x

```
/*
 * Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
```

```
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import java.io.IOException;
import java.util.Collections;
import java.util.Iterator;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.amazon.ion.IonValue;

import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.Owner;
import software.amazon.qldb.tutorial.model.Owners;
import software.amazon.qldb.tutorial.model.Person;
import software.amazon.qldb.tutorial.model.SampleData;

/**
 * Finds and adds secondary owners for a vehicle.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class AddSecondaryOwner {
    public static final Logger log =
 LoggerFactory.getLogger(AddSecondaryOwner.class);

    private AddSecondaryOwner() { }

    /**
     * Check whether a secondary owner has already been registered for the given
 VIN.
     *
     * @param txn
```

```
    *              The {@link TransactionExecutor} for lambda execute.
    * @param vin
    *              Unique VIN for a vehicle.
    * @param secondaryOwnerId
    *              The secondary owner to add.
    * @return {@code true} if the given secondary owner has already been
registered, {@code false} otherwise.
    * @throws IllegalStateException if failed to convert VIN to an {@link
IonValue}.
    */
   public static boolean isSecondaryOwnerForVehicle(final TransactionExecutor
txn, final String vin,
                                                    final String
secondaryOwnerId) {
       try {
           log.info("Finding secondary owners for vehicle with VIN: {}...",
vin);
           final String query = "SELECT Owners.SecondaryOwners FROM
VehicleRegistration AS v WHERE v.VIN = ?";
           final List<IonValue> parameters =
Collections.singletonList(Constants.MAPPER.writeValueAsIonValue(vin));
           final Result result = txn.execute(query, parameters);
           final Iterator<IonValue> itr = result.iterator();
           if (!itr.hasNext()) {
               return false;
           }

           final Owners owners = Constants.MAPPER.readValue(itr.next(),
Owners.class);
           if (null != owners.getSecondaryOwners()) {
               for (Owner owner : owners.getSecondaryOwners()) {
                   if (secondaryOwnerId.equalsIgnoreCase(owner.getPersonId()))
{
                       return true;
                   }
               }
           }

           return false;
       } catch (IOException ioe) {
           throw new IllegalStateException(ioe);
       }
   }
```

```
    /**
     * Adds a secondary owner for the specified VIN.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param vin
     *              Unique VIN for a vehicle.
     * @param secondaryOwner
     *              The secondary owner to add.
     * @throws IllegalStateException if failed to convert parameter into an
{@link IonValue}.
     */
    public static void addSecondaryOwnerForVin(final TransactionExecutor txn,
final String vin,
                                               final String secondaryOwner) {
        try {
            log.info("Inserting secondary owner for vehicle with VIN: {}...",
vin);
            final String query = String.format("FROM VehicleRegistration AS v
WHERE v.VIN = '%s' " +
                    "INSERT INTO v.Owners.SecondaryOwners VALUE ?", vin);
            final IonValue newOwner = Constants.MAPPER.writeValueAsIonValue(new
Owner(secondaryOwner));
            Result result = txn.execute(query, newOwner);
            log.info("VehicleRegistration Document IDs which had secondary
owners added: ");
            ScanTable.printDocuments(result);
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    public static void main(final String... args) {
        final String vin = SampleData.VEHICLES.get(1).getVin();
        final String govId = SampleData.PEOPLE.get(0).getGovId();

        ConnectToLedger.getDriver().execute(txn -> {
            final String documentId = Person.getDocumentIdByGovId(txn, govId);
            if (isSecondaryOwnerForVehicle(txn, vin, documentId)) {
                log.info("Person with ID {} has already been added as a
secondary owner of this vehicle.", govId);
            } else {
                addSecondaryOwnerForVin(txn, vin, documentId);
            }
```

```
        }, (retryAttempt) -> log.info("Retrying due to OCC conflict..."));
        log.info("Secondary owners successfully updated.");
    }
}
```

Pour consulter ces modifications dans le`vehicle-registration` registre, reportez-vous à la sectionÉtape 6 : Afficher l'historique des révisions d'un document.

Étape 6 : Afficher l'historique des révisions d'un document

Après avoir modifié les données d'immatriculation d'un véhicule à l'étape précédente, vous pouvez consulter l'historique de tous ses propriétaires enregistrés et tout autre champ mis à jour. Au cours de cette étape, vous interrogez l'historique des révisions d'un document dans le`VehicleRegistration` tableau de votre`vehicle-registration` registre.

Pour consulter l'historique des révisions

1. Consultez le programme suivant (`QueryHistory.java`).

   2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
```

```
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import java.io.IOException;
import java.time.Instant;
import java.time.temporal.ChronoUnit;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.amazon.ion.IonValue;

import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.SampleData;
import software.amazon.qldb.tutorial.model.VehicleRegistration;

/**
 * Query a table's history for a particular set of documents.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class QueryHistory {
    public static final Logger log =
 LoggerFactory.getLogger(QueryHistory.class);
    private static final int THREE_MONTHS = 90;

    private QueryHistory() { }

    /**
     * In this example, query the 'VehicleRegistration' history table to find
 all previous primary owners for a VIN.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param vin
     *              VIN to find previous primary owners for.
     * @param query
```

```
     *              The query to find previous primary owners.
     * @throws IllegalStateException if failed to convert document ID to an
 {@link IonValue}.
     */
    public static void previousPrimaryOwners(final TransactionExecutor txn,
 final String vin, final String query) {
        try {
            final String docId = VehicleRegistration.getDocumentIdByVin(txn,
 vin);

            log.info("Querying the 'VehicleRegistration' table's history using
 VIN: {}...", vin);
            final Result result = txn.execute(query,
 Constants.MAPPER.writeValueAsIonValue(docId));
            ScanTable.printDocuments(result);
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    public static void main(final String... args) {
        final String threeMonthsAgo = Instant.now().minus(THREE_MONTHS,
 ChronoUnit.DAYS).toString();
        final String query = String.format("SELECT data.Owners.PrimaryOwner,
 metadata.version "
                                           + "FROM history(VehicleRegistration,
 `%s`) "
                                           + "AS h WHERE h.metadata.id = ?",
 threeMonthsAgo);
        ConnectToLedger.getDriver().execute(txn -> {
            final String vin = SampleData.VEHICLES.get(0).getVin();
            previousPrimaryOwners(txn, vin, query);
        });
        log.info("Successfully queried history.");
    }
}
```

1.x

```
/*
 * Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
```

```
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazon.ion.IonValue;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.qldb.QldbSession;
import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.SampleData;
import software.amazon.qldb.tutorial.model.VehicleRegistration;

import java.io.IOException;
import java.time.Instant;
import java.time.temporal.ChronoUnit;
import java.util.Collections;
import java.util.List;

/**
 * Query a table's history for a particular set of documents.
 *
 * This code expects that you have AWS credentials setup per:
```

```
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class QueryHistory {
    public static final Logger log =
 LoggerFactory.getLogger(QueryHistory.class);
    private static final int THREE_MONTHS = 90;

    private QueryHistory() { }

    /**
     * In this example, query the 'VehicleRegistration' history table to find
 all previous primary owners for a VIN.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param vin
     *              VIN to find previous primary owners for.
     * @param query
     *              The query to find previous primary owners.
     * @throws IllegalStateException if failed to convert document ID to an
 {@link IonValue}.
     */
    public static void previousPrimaryOwners(final TransactionExecutor txn,
 final String vin, final String query) {
        try {
            final String docId = VehicleRegistration.getDocumentIdByVin(txn,
 vin);
            final List<IonValue> parameters =
 Collections.singletonList(Constants.MAPPER.writeValueAsIonValue(docId));
            log.info("Querying the 'VehicleRegistration' table's history using
 VIN: {}...", vin);
            final Result result = txn.execute(query, parameters);
            ScanTable.printDocuments(result);
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    public static void main(final String... args) {
        final String threeMonthsAgo = Instant.now().minus(THREE_MONTHS,
 ChronoUnit.DAYS).toString();
        final String query = String.format("SELECT data.Owners.PrimaryOwner,
 metadata.version "
```

```
                                                + "FROM history(VehicleRegistration,
 `%s`) "
                                                + "AS h WHERE h.metadata.id = ?",
 threeMonthsAgo);
        ConnectToLedger.getDriver().execute(txn -> {
            final String vin = SampleData.VEHICLES.get(0).getVin();
            previousPrimaryOwners(txn, vin, query);
        }, (retryAttempt) -> log.info("Retrying due to OCC conflict..."));
        log.info("Successfully queried history.");
    }
}
```

> **Note**
>
> - Vous pouvez consulter l'historique des révisions d'un documentFonction historique en interrogeant la syntaxe intégrée suivante.
>
>   ```
>   SELECT * FROM history( table_name [, `start-time` [, `end-time` ] ] ) AS h
>   [ WHERE h.metadata.id = 'id' ]
>   ```
>
> - L'heure de début et l'heure de fin sont toutes deux facultatives. Il s'agit de valeurs littérales d'Amazon Ion qui peuvent être signalées par des crochets inverses (`` `...` ``). Pour en savoir plus, consultez Interroger Ion avec PartiQL dans Amazon QLDB.
> - Il est recommandé de qualifier une requête d'historique à la fois avec une plage de dates (heure de début et heure de fin) et un identifiant de document (`metadata.id`). QLDB traite les`SELECT` requêtes dans les transactions, qui sont soumises à un délai d'expiration des transactions.
>
>   L'historique QLDB est indexé par ID de document et vous ne pouvez pas créer d'index d'historique supplémentaires pour le moment. Les requêtes d'historique qui incluent une heure de début et une heure de fin bénéficient d'une qualification par plage de dates.

2. Compilez et exécutez le`QueryHistory.java` programme pour interroger l'historique des révisions du`VehicleRegistration` document avec VIN`1N4AL11D75C109151`.

Pour vérifier cryptographiquement la révision d'un document dans le`vehicle-registration` registre, passez àÉtape 7 : Vérification d'un document dans un registre.

Étape 7 : Vérification d'un document dans un registre

Avec Amazon QLDB, vous pouvez vérifier efficacement l'intégrité d'un document dans le journal de votre registre en utilisant le hachage cryptographique SHA-256. Pour en savoir plus sur le fonctionnement de la vérification et du hachage cryptographique dans QLDB, consultez Vérification des données dans Amazon QLDB.

Au cours de cette étape, vous vérifiez la révision d'un document dans le `VehicleRegistration` tableau de votre `vehicle-registration` registre. Tout d'abord, vous demandez un condensé, qui est renvoyé sous forme de fichier de sortie et fait office de signature de l'historique complet des modifications de votre registre. Ensuite, vous demandez une preuve pour la révision par rapport à ce condensé. À l'aide de cette preuve, l'intégrité de votre révision est vérifiée si tous les contrôles de validation sont réussis.

Pour vérifier la révision d'un document

1. Passez en revue les `.java` fichiers suivants, qui représentent les objets QLDB requis pour la vérification et les classes utilitaires avec des méthodes d'assistance pour les valeurs Ion et les chaînes.

   1. `BlockAddress.java`

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
```

```
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.qldb;

import java.util.Objects;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.fasterxml.jackson.annotation.JsonCreator;
import com.fasterxml.jackson.annotation.JsonProperty;

/**
 * Represents the BlockAddress field of a QLDB document.
 */
public final class BlockAddress {

    private static final Logger log =
 LoggerFactory.getLogger(BlockAddress.class);

    private final String strandId;
    private final long sequenceNo;

    @JsonCreator
    public BlockAddress(@JsonProperty("strandId") final String strandId,
                        @JsonProperty("sequenceNo") final long sequenceNo) {
        this.strandId = strandId;
        this.sequenceNo = sequenceNo;
    }

    public long getSequenceNo() {
        return sequenceNo;
    }

    public String getStrandId() {
        return strandId;
    }

    @Override
    public String toString() {
```

```
        return "BlockAddress{"
                + "strandId='" + strandId + '\''
                + ", sequenceNo=" + sequenceNo
                + '}';
    }

    @Override
    public boolean equals(final Object o) {
        if (this == o) {
            return true;
        }
        if (o == null || getClass() != o.getClass()) {
            return false;
        }
        BlockAddress that = (BlockAddress) o;
        return sequenceNo == that.sequenceNo
                && strandId.equals(that.strandId);
    }

    @Override
    public int hashCode() {
        // CHECKSTYLE:OFF - Disabling as we are generating a hashCode of multiple
 properties.
        return Objects.hash(strandId, sequenceNo);
        // CHECKSTYLE:ON
    }
}
```

2. Proof.java

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
```

```
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.qldb;

import com.amazon.ion.IonReader;
import com.amazon.ion.IonSystem;
import com.amazon.ion.system.IonSystemBuilder;
import com.amazonaws.services.qldb.model.GetRevisionRequest;
import com.amazonaws.services.qldb.model.GetRevisionResult;

import java.util.ArrayList;
import java.util.List;

/**
 * A Java representation of the {@link Proof} object.
 * Returned from the {@link
 com.amazonaws.services.qldb.AmazonQLDB#getRevision(GetRevisionRequest)} api.
 */
public final class Proof {
    private static final IonSystem SYSTEM = IonSystemBuilder.standard().build();

    private List<byte[]> internalHashes;

    public Proof(final List<byte[]> internalHashes) {
        this.internalHashes = internalHashes;
    }

    public List<byte[]> getInternalHashes() {
        return internalHashes;
    }

    /**
     * Decodes a {@link Proof} from an ion text String. This ion text is returned
 in
     * a {@link GetRevisionResult#getProof()}
```

```
     *
     * @param ionText
     *              The ion text representing a {@link Proof} object.
     * @return {@link JournalBlock} parsed from the ion text.
     * @throws IllegalStateException if failed to parse the {@link Proof} object
 from the given ion text.
     */
    public static Proof fromBlob(final String ionText) {
        try {
            IonReader reader = SYSTEM.newReader(ionText);
            List<byte[]> list = new ArrayList<>();
            reader.next();
            reader.stepIn();
            while (reader.next() != null) {
                list.add(reader.newBytes());
            }
            return new Proof(list);
        } catch (Exception e) {
            throw new IllegalStateException("Failed to parse a Proof from byte
 array");
        }
    }
}
```

3. QldbIonUtils.java

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
```

```
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.qldb;

import com.amazon.ion.IonReader;
import com.amazon.ion.IonValue;
import com.amazon.ionhash.IonHashReader;
import com.amazon.ionhash.IonHashReaderBuilder;
import com.amazon.ionhash.MessageDigestIonHasherProvider;
import software.amazon.qldb.tutorial.Constants;

public class QldbIonUtils {

    private static MessageDigestIonHasherProvider ionHasherProvider = new
 MessageDigestIonHasherProvider("SHA-256");

    private QldbIonUtils() {}

    /**
     * Builds a hash value from the given {@link IonValue}.
     *
     * @param ionValue
     *              The {@link IonValue} to hash.
     * @return a byte array representing the hash value.
     */
    public static byte[] hashIonValue(final IonValue ionValue) {
        IonReader reader = Constants.SYSTEM.newReader(ionValue);
        IonHashReader hashReader = IonHashReaderBuilder.standard()
                .withHasherProvider(ionHasherProvider)
                .withReader(reader)
                .build();
        while (hashReader.next() != null) {  }
        return hashReader.digest();
    }

}
```

4. QldbStringUtils.java

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.qldb;

import com.amazon.ion.IonWriter;
import com.amazon.ion.system.IonReaderBuilder;
import com.amazon.ion.system.IonTextWriterBuilder;
import com.amazonaws.services.qldb.model.GetBlockResult;
import com.amazonaws.services.qldb.model.GetDigestResult;
import com.amazonaws.services.qldb.model.ValueHolder;

import java.io.IOException;

/**
 * Helper methods to pretty-print certain QLDB response types.
 */
public class QldbStringUtils {

    private QldbStringUtils() {}
```

```
    /**
     * Returns the string representation of a given {@link ValueHolder}.
     * Adapted from the AWS SDK autogenerated {@code toString()} method, with
sensitive values un-redacted.
     * Additionally, this method pretty-prints any IonText included in the {@link
ValueHolder}.
     *
     * @param valueHolder the {@link ValueHolder} to convert to a String.
     * @return the String representation of the supplied {@link ValueHolder}.
     */
    public static String toUnredactedString(ValueHolder valueHolder) {
        StringBuilder sb = new StringBuilder();
        sb.append("{");
        if (valueHolder.getIonText() != null) {

            sb.append("IonText: ");
            IonWriter prettyWriter = IonTextWriterBuilder.pretty().build(sb);
            try {

prettyWriter.writeValues(IonReaderBuilder.standard().build(valueHolder.getIonText()));
            } catch (IOException ioe) {
                sb.append("**Exception while printing this IonText**");
            }
        }

        sb.append("}");
        return sb.toString();
    }

    /**
     * Returns the string representation of a given {@link GetBlockResult}.
     * Adapted from the AWS SDK autogenerated {@code toString()} method, with
sensitive values un-redacted.
     *
     * @param getBlockResult the {@link GetBlockResult} to convert to a String.
     * @return the String representation of the supplied {@link GetBlockResult}.
     */
    public static String toUnredactedString(GetBlockResult getBlockResult) {
        StringBuilder sb = new StringBuilder();
        sb.append("{");
        if (getBlockResult.getBlock() != null) {
            sb.append("Block:
").append(toUnredactedString(getBlockResult.getBlock())).append(",");
        }
```

```
        if (getBlockResult.getProof() != null) {
            sb.append("Proof:
").append(toUnredactedString(getBlockResult.getProof()));
        }

        sb.append("}");
        return sb.toString();
    }

    /**
     * Returns the string representation of a given {@link GetDigestResult}.
     * Adapted from the AWS SDK autogenerated {@code toString()} method, with
 sensitive values un-redacted.
     *
     * @param getDigestResult the {@link GetDigestResult} to convert to a String.
     * @return the String representation of the supplied {@link GetDigestResult}.
     */
    public static String toUnredactedString(GetDigestResult getDigestResult) {
        StringBuilder sb = new StringBuilder();
        sb.append("{");
        if (getDigestResult.getDigest() != null) {
            sb.append("Digest:
").append(getDigestResult.getDigest()).append(",");
        }

        if (getDigestResult.getDigestTipAddress() != null) {
            sb.append("DigestTipAddress:
").append(toUnredactedString(getDigestResult.getDigestTipAddress()));
        }

        sb.append("}");
        return sb.toString();
    }
}
```

5. `Verifier.java`

   2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
```

```
 * Permission is hereby granted, free of charge, to any person obtaining a
 copy of this
 * software and associated documentation files (the "Software"), to deal in
 the Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR
 A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Comparator;
import java.util.Iterator;
import java.util.List;
import java.util.concurrent.ThreadLocalRandom;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.amazonaws.util.Base64;

import software.amazon.qldb.tutorial.qldb.Proof;

/**
```

```
 * Encapsulates the logic to verify the integrity of revisions or blocks in a
 QLDB ledger.
 *
 * The main entry point is {@link #verify(byte[], byte[], String)}.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class Verifier {
    public static final Logger log = LoggerFactory.getLogger(Verifier.class);
    private static final int HASH_LENGTH = 32;
    private static final int UPPER_BOUND = 8;

    /**
     * Compares two hashes by their <em>signed</em> byte values in little-
endian order.
     */
    private static Comparator<byte[]> hashComparator = (h1, h2) -> {
        if (h1.length != HASH_LENGTH || h2.length != HASH_LENGTH) {
            throw new IllegalArgumentException("Invalid hash.");
        }
        for (int i = h1.length - 1; i >= 0; i--) {
            int byteEqual = Byte.compare(h1[i], h2[i]);
            if (byteEqual != 0) {
                return byteEqual;
            }
        }

        return 0;
    };

    private Verifier() { }

    /**
     * Verify the integrity of a document with respect to a QLDB ledger
 digest.
     *
     * The verification algorithm includes the following steps:
     *
     * 1. {@link #buildCandidateDigest(Proof, byte[])} build the candidate
 digest from the internal hashes
     * in the {@link Proof}.
```

```
    * 2. Check that the {@code candidateLedgerDigest} is equal to the {@code
ledgerDigest}.
    *
    * @param documentHash
    *              The hash of the document to be verified.
    * @param digest
    *              The QLDB ledger digest. This digest should have been
retrieved using
    *              {@link com.amazonaws.services.qldb.AmazonQLDB#getDigest}
    * @param proofBlob
    *              The ion encoded bytes representing the {@link Proof}
associated with the supplied
    *              {@code digestTipAddress} and {@code address} retrieved
using
    *              {@link
com.amazonaws.services.qldb.AmazonQLDB#getRevision}.
    * @return {@code true} if the record is verified or {@code false} if it
is not verified.
    */
   public static boolean verify(
           final byte[] documentHash,
           final byte[] digest,
           final String proofBlob
   ) {
       Proof proof = Proof.fromBlob(proofBlob);

       byte[] candidateDigest = buildCandidateDigest(proof, documentHash);

       return Arrays.equals(digest, candidateDigest);
   }

   /**
    * Build the candidate digest representing the entire ledger from the
internal hashes of the {@link Proof}.
    *
    * @param proof
    *              A Java representation of {@link Proof}
    *              returned from {@link
com.amazonaws.services.qldb.AmazonQLDB#getRevision}.
    * @param leafHash
    *              Leaf hash to build the candidate digest with.
    * @return a byte array of the candidate digest.
    */
```

```
   private static byte[] buildCandidateDigest(final Proof proof, final byte[]
leafHash) {
       return calculateRootHashFromInternalHashes(proof.getInternalHashes(),
leafHash);
   }

   /**
    * Get a new instance of {@link MessageDigest} using the SHA-256
algorithm.
    *
    * @return an instance of {@link MessageDigest}.
    * @throws IllegalStateException if the algorithm is not available on the
current JVM.
    */
   static MessageDigest newMessageDigest() {
       try {
           return MessageDigest.getInstance("SHA-256");
       } catch (NoSuchAlgorithmException e) {
           log.error("Failed to create SHA-256 MessageDigest", e);
           throw new IllegalStateException("SHA-256 message digest is
unavailable", e);
       }
   }

   /**
    * Takes two hashes, sorts them, concatenates them, and then returns the
    * hash of the concatenated array.
    *
    * @param h1
    *              Byte array containing one of the hashes to compare.
    * @param h2
    *              Byte array containing one of the hashes to compare.
    * @return the concatenated array of hashes.
    */
   public static byte[] dot(final byte[] h1, final byte[] h2) {
       if (h1.length == 0) {
           return h2;
       }
       if (h2.length == 0) {
           return h1;
       }
       byte[] concatenated = new byte[h1.length + h2.length];
       if (hashComparator.compare(h1, h2) < 0) {
           System.arraycopy(h1, 0, concatenated, 0, h1.length);
```

```
            System.arraycopy(h2, 0, concatenated, h1.length, h2.length);
        } else {
            System.arraycopy(h2, 0, concatenated, 0, h2.length);
            System.arraycopy(h1, 0, concatenated, h2.length, h1.length);
        }
        MessageDigest messageDigest = newMessageDigest();
        messageDigest.update(concatenated);

        return messageDigest.digest();
    }

    /**
     * Starting with the provided {@code leafHash} combined with the provided
{@code internalHashes}
     * pairwise until only the root hash remains.
     *
     * @param internalHashes
     *              Internal hashes of Merkle tree.
     * @param leafHash
     *              Leaf hashes of Merkle tree.
     * @return the root hash.
     */
    private static byte[] calculateRootHashFromInternalHashes(final
List<byte[]> internalHashes, final byte[] leafHash) {
        return internalHashes.stream().reduce(leafHash, Verifier::dot);
    }

    /**
     * Flip a single random bit in the given byte array. This method is used
to demonstrate
     * QLDB's verification features.
     *
     * @param original
     *              The original byte array.
     * @return the altered byte array with a single random bit changed.
     */
    public static byte[] flipRandomBit(final byte[] original) {
        if (original.length == 0) {
            throw new IllegalArgumentException("Array cannot be empty!");
        }
        int alteredPosition =
ThreadLocalRandom.current().nextInt(original.length);
        int b = ThreadLocalRandom.current().nextInt(UPPER_BOUND);
        byte[] altered = new byte[original.length];
```

```
        System.arraycopy(original, 0, altered, 0, original.length);
        altered[alteredPosition] = (byte) (altered[alteredPosition] ^ (1 <<
b));
        return altered;
    }

    public static String toBase64(byte[] arr) {
        return new String(Base64.encode(arr), StandardCharsets.UTF_8);
    }

    /**
     * Convert a {@link ByteBuffer} into byte array.
     *
     * @param buffer
     *              The {@link ByteBuffer} to convert.
     * @return the converted byte array.
     */
    public static byte[] convertByteBufferToByteArray(final ByteBuffer buffer)
{
        byte[] arr = new byte[buffer.remaining()];
        buffer.get(arr);
        return arr;
    }

    /**
     * Calculates the root hash from a list of hashes that represent the base
of a Merkle tree.
     *
     * @param hashes
     *              The list of byte arrays representing hashes making up base
of a Merkle tree.
     * @return a byte array that is the root hash of the given list of hashes.
     */
    public static byte[] calculateMerkleTreeRootHash(List<byte[]> hashes) {
        if (hashes.isEmpty()) {
            return new byte[0];
        }

        List<byte[]> remaining = combineLeafHashes(hashes);
        while (remaining.size() > 1) {
            remaining = combineLeafHashes(remaining);
        }
        return remaining.get(0);
    }
```

```
    private static List<byte[]> combineLeafHashes(List<byte[]> hashes) {
        List<byte[]> combinedHashes = new ArrayList<>();
        Iterator<byte[]> it = hashes.stream().iterator();

        while (it.hasNext()) {
            byte[] left = it.next();
            if (it.hasNext()) {
                byte[] right = it.next();
                byte[] combined = dot(left, right);
                combinedHashes.add(combined);
            } else {
                combinedHashes.add(left);
            }
        }

        return combinedHashes;
    }
}
```

1.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
 copy of this
 * software and associated documentation files (the "Software"), to deal in
 the Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR
 A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
```

```
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazonaws.util.Base64;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.qldb.tutorial.qldb.Proof;

import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.*;
import java.util.concurrent.ThreadLocalRandom;

/**
 * Encapsulates the logic to verify the integrity of revisions or blocks in a
 QLDB ledger.
 *
 * The main entry point is {@link #verify(byte[], byte[], String)}.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class Verifier {
    public static final Logger log = LoggerFactory.getLogger(Verifier.class);
    private static final int HASH_LENGTH = 32;
    private static final int UPPER_BOUND = 8;

    /**
     * Compares two hashes by their <em>signed</em> byte values in little-
endian order.
     */
    private static Comparator<byte[]> hashComparator = (h1, h2) -> {
        if (h1.length != HASH_LENGTH || h2.length != HASH_LENGTH) {
            throw new IllegalArgumentException("Invalid hash.");
        }
```

```
        for (int i = h1.length - 1; i >= 0; i--) {
            int byteEqual = Byte.compare(h1[i], h2[i]);
            if (byteEqual != 0) {
                return byteEqual;
            }
        }

        return 0;
    };

    private Verifier() { }

    /**
     * Verify the integrity of a document with respect to a QLDB ledger
digest.
     *
     * The verification algorithm includes the following steps:
     *
     * 1. {@link #buildCandidateDigest(Proof, byte[])} build the candidate
digest from the internal hashes
     * in the {@link Proof}.
     * 2. Check that the {@code candidateLedgerDigest} is equal to the {@code
ledgerDigest}.
     *
     * @param documentHash
     *              The hash of the document to be verified.
     * @param digest
     *              The QLDB ledger digest. This digest should have been
retrieved using
     *              {@link com.amazonaws.services.qldb.AmazonQLDB#getDigest}
     * @param proofBlob
     *              The ion encoded bytes representing the {@link Proof}
associated with the supplied
     *              {@code digestTipAddress} and {@code address} retrieved
using
     *              {@link
com.amazonaws.services.qldb.AmazonQLDB#getRevision}.
     * @return {@code true} if the record is verified or {@code false} if it
is not verified.
     */
    public static boolean verify(
            final byte[] documentHash,
            final byte[] digest,
            final String proofBlob
```

```
    ) {
        Proof proof = Proof.fromBlob(proofBlob);

        byte[] candidateDigest = buildCandidateDigest(proof, documentHash);

        return Arrays.equals(digest, candidateDigest);
    }

    /**
     * Build the candidate digest representing the entire ledger from the
internal hashes of the {@link Proof}.
     *
     * @param proof
     *              A Java representation of {@link Proof}
     *              returned from {@link
com.amazonaws.services.qldb.AmazonQLDB#getRevision}.
     * @param leafHash
     *              Leaf hash to build the candidate digest with.
     * @return a byte array of the candidate digest.
     */
    private static byte[] buildCandidateDigest(final Proof proof, final byte[]
leafHash) {
        return calculateRootHashFromInternalHashes(proof.getInternalHashes(),
leafHash);
    }

    /**
     * Get a new instance of {@link MessageDigest} using the SHA-256
algorithm.
     *
     * @return an instance of {@link MessageDigest}.
     * @throws IllegalStateException if the algorithm is not available on the
current JVM.
     */
    static MessageDigest newMessageDigest() {
        try {
            return MessageDigest.getInstance("SHA-256");
        } catch (NoSuchAlgorithmException e) {
            log.error("Failed to create SHA-256 MessageDigest", e);
            throw new IllegalStateException("SHA-256 message digest is
unavailable", e);
        }
    }
```

```
    /**
     * Takes two hashes, sorts them, concatenates them, and then returns the
     * hash of the concatenated array.
     *
     * @param h1
     *              Byte array containing one of the hashes to compare.
     * @param h2
     *              Byte array containing one of the hashes to compare.
     * @return the concatenated array of hashes.
     */
    public static byte[] dot(final byte[] h1, final byte[] h2) {
        if (h1.length == 0) {
            return h2;
        }
        if (h2.length == 0) {
            return h1;
        }
        byte[] concatenated = new byte[h1.length + h2.length];
        if (hashComparator.compare(h1, h2) < 0) {
            System.arraycopy(h1, 0, concatenated, 0, h1.length);
            System.arraycopy(h2, 0, concatenated, h1.length, h2.length);
        } else {
            System.arraycopy(h2, 0, concatenated, 0, h2.length);
            System.arraycopy(h1, 0, concatenated, h2.length, h1.length);
        }
        MessageDigest messageDigest = newMessageDigest();
        messageDigest.update(concatenated);

        return messageDigest.digest();
    }

    /**
     * Starting with the provided {@code leafHash} combined with the provided
{@code internalHashes}
     * pairwise until only the root hash remains.
     *
     * @param internalHashes
     *              Internal hashes of Merkle tree.
     * @param leafHash
     *              Leaf hashes of Merkle tree.
     * @return the root hash.
     */
    private static byte[] calculateRootHashFromInternalHashes(final
List<byte[]> internalHashes, final byte[] leafHash) {
```

```
        return internalHashes.stream().reduce(leafHash, Verifier::dot);
    }

    /**
     * Flip a single random bit in the given byte array. This method is used
to demonstrate
     * QLDB's verification features.
     *
     * @param original
     *              The original byte array.
     * @return the altered byte array with a single random bit changed.
     */
    public static byte[] flipRandomBit(final byte[] original) {
        if (original.length == 0) {
            throw new IllegalArgumentException("Array cannot be empty!");
        }
        int alteredPosition =
ThreadLocalRandom.current().nextInt(original.length);
        int b = ThreadLocalRandom.current().nextInt(UPPER_BOUND);
        byte[] altered = new byte[original.length];
        System.arraycopy(original, 0, altered, 0, original.length);
        altered[alteredPosition] = (byte) (altered[alteredPosition] ^ (1 <<
b));
        return altered;
    }

    public static String toBase64(byte[] arr) {
        return new String(Base64.encode(arr), StandardCharsets.UTF_8);
    }

    /**
     * Convert a {@link ByteBuffer} into byte array.
     *
     * @param buffer
     *              The {@link ByteBuffer} to convert.
     * @return the converted byte array.
     */
    public static byte[] convertByteBufferToByteArray(final ByteBuffer buffer)
{
        byte[] arr = new byte[buffer.remaining()];
        buffer.get(arr);
        return arr;
    }
```

```
    /**
     * Calculates the root hash from a list of hashes that represent the base
 of a Merkle tree.
     *
     * @param hashes
     *              The list of byte arrays representing hashes making up base
 of a Merkle tree.
     * @return a byte array that is the root hash of the given list of hashes.
     */
    public static byte[] calculateMerkleTreeRootHash(List<byte[]> hashes) {
        if (hashes.isEmpty()) {
            return new byte[0];
        }

        List<byte[]> remaining = combineLeafHashes(hashes);
        while (remaining.size() > 1) {
            remaining = combineLeafHashes(remaining);
        }
        return remaining.get(0);
    }

    private static List<byte[]> combineLeafHashes(List<byte[]> hashes) {
        List<byte[]> combinedHashes = new ArrayList<>();
        Iterator<byte[]> it = hashes.stream().iterator();

        while (it.hasNext()) {
            byte[] left = it.next();
            if (it.hasNext()) {
                byte[] right = it.next();
                byte[] combined = dot(left, right);
                combinedHashes.add(combined);
            } else {
                combinedHashes.add(left);
            }
        }

        return combinedHashes;
    }
}
```

2. Utilisez deux `.java` fichiers (`GetDigest.javaetGetRevision.java`) pour effectuer les opérations suivantes :

   - Demandez un nouveau résumé depuis le`vehicle-registration` registre.

- Demandez une preuve pour chaque révision d'un document à partir du`VehicleRegistration` tableau.
- Vérifiez les révisions à l'aide du condensé renvoyé et corrigez-le en recalculant le résumé.

Le`GetDigest.java` programme contient le code suivant.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazonaws.services.qldb.AmazonQLDB;
import com.amazonaws.services.qldb.model.GetDigestRequest;
import com.amazonaws.services.qldb.model.GetDigestResult;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.qldb.tutorial.qldb.QldbStringUtils;

/**
```

```
 * This is an example for retrieving the digest of a particular ledger.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class GetDigest {
    public static final Logger log = LoggerFactory.getLogger(GetDigest.class);
    public static AmazonQLDB client = CreateLedger.getClient();

    private GetDigest() { }

    /**
     * Calls {@link #getDigest(String)} for a ledger.
     *
     * @param args
     *              Arbitrary command-line arguments.
     * @throws Exception if failed to get a ledger digest.
     */
    public static void main(final String... args) throws Exception {
        try {

            getDigest(Constants.LEDGER_NAME);

        } catch (Exception e) {
            log.error("Unable to get a ledger digest!", e);
            throw e;
        }
    }

    /**
     * Get the digest for the specified ledger.
     *
     * @param ledgerName
     *              The ledger to get digest from.
     * @return {@link GetDigestResult}.
     */
    public static GetDigestResult getDigest(final String ledgerName) {
        log.info("Let's get the current digest of the ledger named {}.",
 ledgerName);
        GetDigestRequest request = new GetDigestRequest()
                .withName(ledgerName);
        GetDigestResult result = client.getDigest(request);
```

```
        log.info("Success. LedgerDigest: {}.",
 QldbStringUtils.toUnredactedString(result));
        return result;
    }
}
```

> **Note**
>
> Utilisez`getDigest` cette méthode pour demander un résumé reprenant la partie actuelle du journal dans votre grand livre. La pointe du journal fait référence au dernier bloc validé au moment où QLDB reçoit votre demande.

`LeGetRevision.java` programme contient le code suivant.

2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */
```

```
package software.amazon.qldb.tutorial;

import com.amazon.ion.IonReader;
import com.amazon.ion.IonStruct;
import com.amazon.ion.IonSystem;
import com.amazon.ion.IonValue;
import com.amazon.ion.IonWriter;
import com.amazon.ion.system.IonReaderBuilder;
import com.amazon.ion.system.IonSystemBuilder;
import com.amazonaws.services.qldb.AmazonQLDB;
import com.amazonaws.services.qldb.model.GetDigestResult;
import com.amazonaws.services.qldb.model.GetRevisionRequest;
import com.amazonaws.services.qldb.model.GetRevisionResult;
import com.amazonaws.services.qldb.model.ValueHolder;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.qldb.QldbDriver;
import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.SampleData;
import software.amazon.qldb.tutorial.qldb.BlockAddress;
import software.amazon.qldb.tutorial.qldb.QldbRevision;
import software.amazon.qldb.tutorial.qldb.QldbStringUtils;

/**
 * Verify the integrity of a document revision in a QLDB ledger.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class GetRevision {
    public static final Logger log = LoggerFactory.getLogger(GetRevision.class);
    public static AmazonQLDB client = CreateLedger.getClient();
    private static final IonSystem SYSTEM = IonSystemBuilder.standard().build();

    private GetRevision() { }
```

```
    public static void main(String... args) throws Exception {

        final String vin = SampleData.REGISTRATIONS.get(0).getVin();


        verifyRegistration(ConnectToLedger.getDriver(), Constants.LEDGER_NAME,
 vin);
    }

    /**
     * Verify each version of the registration for the given VIN.
     *
     * @param driver
     *              A QLDB driver.
     * @param ledgerName
     *              The ledger to get digest from.
     * @param vin
     *              VIN to query the revision history of a specific registration
 with.
     * @throws Exception if failed to verify digests.
     * @throws AssertionError if document revision verification failed.
     */
    public static void verifyRegistration(final QldbDriver driver, final String
 ledgerName, final String vin)
            throws Exception {
        log.info(String.format("Let's verify the registration with VIN=%s, in
 ledger=%s.", vin, ledgerName));

        try {
            log.info("First, let's get a digest.");
            GetDigestResult digestResult = GetDigest.getDigest(ledgerName);

            ValueHolder digestTipAddress = digestResult.getDigestTipAddress();
            byte[] digestBytes =
 Verifier.convertByteBufferToByteArray(digestResult.getDigest());

            log.info("Got a ledger digest. Digest end address={}, digest={}.",
                QldbStringUtils.toUnredactedString(digestTipAddress),
                Verifier.toBase64(digestBytes));

            log.info(String.format("Next, let's query the registration with VIN=
%s. "
                    + "Then we can verify each version of the registration.",
 vin));
```

```
            List<IonStruct> documentsWithMetadataList = new ArrayList<>();
            driver.execute(txn -> {
                documentsWithMetadataList.addAll(queryRegistrationsByVin(txn,
vin));
            });
            log.info("Registrations queried successfully!");

            log.info(String.format("Found %s revisions of the registration with
VIN=%s.",
                    documentsWithMetadataList.size(), vin));

            for (IonStruct ionStruct : documentsWithMetadataList) {

                QldbRevision document = QldbRevision.fromIon(ionStruct);
                log.info(String.format("Let's verify the document: %s",
document));

                log.info("Let's get a proof for the document.");
                GetRevisionResult proofResult = getRevision(
                        ledgerName,
                        document.getMetadata().getId(),
                        digestTipAddress,
                        document.getBlockAddress()
                );

                final IonValue proof =
Constants.MAPPER.writeValueAsIonValue(proofResult.getProof());
                final IonReader reader =
IonReaderBuilder.standard().build(proof);
                reader.next();
                ByteArrayOutputStream baos = new ByteArrayOutputStream();
                IonWriter writer = SYSTEM.newBinaryWriter(baos);
                writer.writeValue(reader);
                writer.close();
                baos.flush();
                baos.close();
                byte[] byteProof = baos.toByteArray();

                log.info(String.format("Got back a proof: %s",
Verifier.toBase64(byteProof)));

                boolean verified = Verifier.verify(
                        document.getHash(),
                        digestBytes,
```

```
                        proofResult.getProof().getIonText()
                );

                if (!verified) {
                    throw new AssertionError("Document revision is not
verified!");
                } else {
                    log.info("Success! The document is verified");
                }

                byte[] alteredDigest = Verifier.flipRandomBit(digestBytes);
                log.info(String.format("Flipping one bit in the digest and
assert that the document is NOT verified. "
                        + "The altered digest is: %s",
Verifier.toBase64(alteredDigest)));
                verified = Verifier.verify(
                        document.getHash(),
                        alteredDigest,
                        proofResult.getProof().getIonText()
                );

                if (verified) {
                    throw new AssertionError("Expected document to not be
verified against altered digest.");
                } else {
                    log.info("Success! As expected flipping a bit in the digest
causes verification to fail.");
                }

                byte[] alteredDocumentHash =
Verifier.flipRandomBit(document.getHash());
                log.info(String.format("Flipping one bit in the document's hash
and assert that it is NOT verified. "
                        + "The altered document hash is: %s.",
Verifier.toBase64(alteredDocumentHash)));
                verified = Verifier.verify(
                        alteredDocumentHash,
                        digestBytes,
                        proofResult.getProof().getIonText()
                );

                if (verified) {
                    throw new AssertionError("Expected altered document hash to
not be verified against digest.");
```

```
                } else {
                    log.info("Success! As expected flipping a bit in the
document hash causes verification to fail.");
                }
            }

        } catch (Exception e) {
            log.error("Failed to verify digests.", e);
            throw e;
        }

        log.info(String.format("Finished verifying the registration with VIN=%s
in ledger=%s.", vin, ledgerName));
    }

    /**
     * Get the revision of a particular document specified by the given document
ID and block address.
     *
     * @param ledgerName
     *              Name of the ledger containing the document.
     * @param documentId
     *              Unique ID for the document to be verified, contained in the
committed view of the document.
     * @param digestTipAddress
     *              The latest block location covered by the digest.
     * @param blockAddress
     *              The location of the block to request.
     * @return the requested revision.
     */
    public static GetRevisionResult getRevision(final String ledgerName, final
String documentId,
                                                final ValueHolder
digestTipAddress, final BlockAddress blockAddress) {
        try {
            GetRevisionRequest request = new GetRevisionRequest()
                    .withName(ledgerName)
                    .withDigestTipAddress(digestTipAddress)
                    .withBlockAddress(new
ValueHolder().withIonText(Constants.MAPPER.writeValueAsIonValue(blockAddress)
                            .toString()))
                    .withDocumentId(documentId);
            return client.getRevision(request);
        } catch (IOException ioe) {
```

```
            throw new IllegalStateException(ioe);
        }
    }

    /**
     * Query the registration history for the given VIN.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param vin
     *              The unique VIN to query.
     * @return a list of {@link IonStruct} representing the registration
 history.
     * @throws IllegalStateException if failed to convert parameters into {@link
 IonValue}
     */
    public static List<IonStruct> queryRegistrationsByVin(final
 TransactionExecutor txn, final String vin) {
        log.info(String.format("Let's query the 'VehicleRegistration' table for
 VIN: %s...", vin));
        log.info("Let's query the 'VehicleRegistration' table for VIN: {}...",
 vin);
        final String query = String.format("SELECT * FROM _ql_committed_%s WHERE
 data.VIN = ?",
                Constants.VEHICLE_REGISTRATION_TABLE_NAME);
        try {
            final List<IonValue> parameters =
 Collections.singletonList(Constants.MAPPER.writeValueAsIonValue(vin));
            final Result result = txn.execute(query, parameters);
            List<IonStruct> list = ScanTable.toIonStructs(result);
            log.info(String.format("Found %d document(s)!", list.size()));
            return list;
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }
}
```

1.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
```

```
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazon.ion.IonReader;
import com.amazon.ion.IonStruct;
import com.amazon.ion.IonValue;
import com.amazon.ion.IonWriter;
import com.amazon.ion.system.IonReaderBuilder;
import com.amazonaws.services.qldb.AmazonQLDB;
import com.amazonaws.services.qldb.model.GetDigestResult;
import com.amazonaws.services.qldb.model.GetRevisionRequest;
import com.amazonaws.services.qldb.model.GetRevisionResult;
import com.amazonaws.services.qldb.model.ValueHolder;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.qldb.QldbSession;
import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.SampleData;
import software.amazon.qldb.tutorial.qldb.BlockAddress;
import software.amazon.qldb.tutorial.qldb.QldbRevision;
import software.amazon.qldb.tutorial.qldb.QldbStringUtils;
```

```
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

/**
 * Verify the integrity of a document revision in a QLDB ledger.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class GetRevision {
    public static final Logger log = LoggerFactory.getLogger(GetRevision.class);
    public static AmazonQLDB client = CreateLedger.getClient();

    private GetRevision() { }

    public static void main(String... args) throws Exception {

        final String vin = SampleData.REGISTRATIONS.get(0).getVin();

        try (QldbSession qldbSession = ConnectToLedger.createQldbSession()) {
            verifyRegistration(qldbSession, Constants.LEDGER_NAME, vin);
        }
    }

    /**
     * Verify each version of the registration for the given VIN.
     *
     * @param qldbSession
     *              A QLDB session.
     * @param ledgerName
     *              The ledger to get digest from.
     * @param vin
     *              VIN to query the revision history of a specific registration
 with.
     * @throws Exception if failed to verify digests.
     * @throws AssertionError if document revision verification failed.
     */
    public static void verifyRegistration(final QldbSession qldbSession, final
 String ledgerName, final String vin)
```

```
            throws Exception {
        log.info(String.format("Let's verify the registration with VIN=%s, in
 ledger=%s.", vin, ledgerName));

        try {
            log.info("First, let's get a digest.");
            GetDigestResult digestResult = GetDigest.getDigest(ledgerName);

            ValueHolder digestTipAddress = digestResult.getDigestTipAddress();
            byte[] digestBytes =
 Verifier.convertByteBufferToByteArray(digestResult.getDigest());

            log.info("Got a ledger digest. Digest end address={}, digest={}.",
                QldbStringUtils.toUnredactedString(digestTipAddress),
                Verifier.toBase64(digestBytes));

            log.info(String.format("Next, let's query the registration with VIN=
%s. "
                    + "Then we can verify each version of the registration.",
 vin));
            List<IonStruct> documentsWithMetadataList = new ArrayList<>();
            qldbSession.execute(txn -> {
                documentsWithMetadataList.addAll(queryRegistrationsByVin(txn,
 vin));
            }, (retryAttempt) -> log.info("Retrying due to OCC conflict..."));
            log.info("Registrations queried successfully!");

            log.info(String.format("Found %s revisions of the registration with
 VIN=%s.",
                    documentsWithMetadataList.size(), vin));

            for (IonStruct ionStruct : documentsWithMetadataList) {

                QldbRevision document = QldbRevision.fromIon(ionStruct);
                log.info(String.format("Let's verify the document: %s",
 document));

                log.info("Let's get a proof for the document.");
                GetRevisionResult proofResult = getRevision(
                        ledgerName,
                        document.getMetadata().getId(),
                        digestTipAddress,
                        document.getBlockAddress()
                );
```

```
                final IonValue proof =
Constants.MAPPER.writeValueAsIonValue(proofResult.getProof());
                final IonReader reader =
IonReaderBuilder.standard().build(proof);
                reader.next();
                ByteArrayOutputStream baos = new ByteArrayOutputStream();
                IonWriter writer = Constants.SYSTEM.newBinaryWriter(baos);
                writer.writeValue(reader);
                writer.close();
                baos.flush();
                baos.close();
                byte[] byteProof = baos.toByteArray();

                log.info(String.format("Got back a proof: %s",
Verifier.toBase64(byteProof)));

                boolean verified = Verifier.verify(
                        document.getHash(),
                        digestBytes,
                        proofResult.getProof().getIonText()
                );

                if (!verified) {
                    throw new AssertionError("Document revision is not
verified!");
                } else {
                    log.info("Success! The document is verified");
                }

                byte[] alteredDigest = Verifier.flipRandomBit(digestBytes);
                log.info(String.format("Flipping one bit in the digest and
assert that the document is NOT verified. "
                        + "The altered digest is: %s",
Verifier.toBase64(alteredDigest)));
                verified = Verifier.verify(
                        document.getHash(),
                        alteredDigest,
                        proofResult.getProof().getIonText()
                );

                if (verified) {
                    throw new AssertionError("Expected document to not be
verified against altered digest.");
```

```
                } else {
                    log.info("Success! As expected flipping a bit in the digest
causes verification to fail.");
                }

                byte[] alteredDocumentHash =
Verifier.flipRandomBit(document.getHash());
                log.info(String.format("Flipping one bit in the document's hash
and assert that it is NOT verified. "
                        + "The altered document hash is: %s.",
Verifier.toBase64(alteredDocumentHash)));
                verified = Verifier.verify(
                        alteredDocumentHash,
                        digestBytes,
                        proofResult.getProof().getIonText()
                );

                if (verified) {
                    throw new AssertionError("Expected altered document hash to
not be verified against digest.");
                } else {
                    log.info("Success! As expected flipping a bit in the
document hash causes verification to fail.");
                }
            }

        } catch (Exception e) {
            log.error("Failed to verify digests.", e);
            throw e;
        }

        log.info(String.format("Finished verifying the registration with VIN=%s
in ledger=%s.", vin, ledgerName));
    }

    /**
     * Get the revision of a particular document specified by the given document
ID and block address.
     *
     * @param ledgerName
     *              Name of the ledger containing the document.
     * @param documentId
     *              Unique ID for the document to be verified, contained in the
committed view of the document.
```

```
    * @param digestTipAddress
    *              The latest block location covered by the digest.
    * @param blockAddress
    *              The location of the block to request.
    * @return the requested revision.
    */
   public static GetRevisionResult getRevision(final String ledgerName, final
String documentId,
                                               final ValueHolder
digestTipAddress, final BlockAddress blockAddress) {
       try {
           GetRevisionRequest request = new GetRevisionRequest()
                   .withName(ledgerName)
                   .withDigestTipAddress(digestTipAddress)
                   .withBlockAddress(new
ValueHolder().withIonText(Constants.MAPPER.writeValueAsIonValue(blockAddress)
                           .toString()))
                   .withDocumentId(documentId);
           return client.getRevision(request);
       } catch (IOException ioe) {
           throw new IllegalStateException(ioe);
       }
   }

   /**
    * Query the registration history for the given VIN.
    *
    * @param txn
    *              The {@link TransactionExecutor} for lambda execute.
    * @param vin
    *              The unique VIN to query.
    * @return a list of {@link IonStruct} representing the registration
history.
    * @throws IllegalStateException if failed to convert parameters into {@link
IonValue}
    */
   public static List<IonStruct> queryRegistrationsByVin(final
TransactionExecutor txn, final String vin) {
       log.info(String.format("Let's query the 'VehicleRegistration' table for
VIN: %s...", vin));
       log.info("Let's query the 'VehicleRegistration' table for VIN: {}...",
vin);
       final String query = String.format("SELECT * FROM _ql_committed_%s WHERE
data.VIN = ?",
```

```
                Constants.VEHICLE_REGISTRATION_TABLE_NAME);
        try {
            final List<IonValue> parameters =
 Collections.singletonList(Constants.MAPPER.writeValueAsIonValue(vin));
            final Result result = txn.execute(query, parameters);
            List<IonStruct> list = ScanTable.toIonStructs(result);
            log.info(String.format("Found %d document(s)!", list.size()));
            return list;
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }
}
```

> **Note**
>
> Une fois que la`getRevision` méthode a renvoyé une preuve pour la révision du document spécifiée, ce programme utilise une API côté client pour vérifier cette révision. Pour obtenir une vue d'ensemble de l'algorithme utilisé par cette API, consultezUtiliser une preuve pour recalculer votre résumé.

3. Compilez et exécutez le`GetRevision.java` programme pour vérifier cryptographiquement le`VehicleRegistration` document avec le VIN`1N4AL11D75C109151`.

Pour exporter et valider les données du journal dans le`vehicle-registration` grand livre, passez àÉtape 8 : Exporter et valider les données du journal dans un registre.

Étape 8 : Exporter et valider les données du journal dans un registre

Dans Amazon QLDB, vous pouvez accéder au contenu du journal dans votre grand livre à des fins diverses, telles que la conservation des données, l'analyse et l'audit. Pour plus d'informations, veuillez consulter Exportation de données de journal depuis Amazon QLDB.

Au cours de cette étape, vous exportez des blocs de journal depuis le`vehicle-registration` registre vers un compartiment Amazon S3. Vous utilisez ensuite les données exportées pour valider la chaîne de hachage entre les blocs de journal et les composants de hachage individuels de chaque bloc.

L'entitéAWS Identity and Access Management (IAM) principale que vous utilisez doit disposer d'autorisations IAM suffisantes pour créer un compartiment Amazon S3 dans votreCompte AWS. Pour plus d'informations, consultez les politiques et autorisations d'Amazon S3 dans le guide de l'utilisateur d'Amazon S3. Vous devez également disposer des autorisations nécessaires pour créer un rôle IAM assorti d'une politique d'autorisations permettant à QLDB d'écrire des objets dans votre compartiment Amazon S3. Pour en savoir plus, consultez la section Autorisations requises pour accéder aux ressources IAM dans le Guide de l'utilisateur IAM.

Pour exporter et valider les données du journal

1. Consultez le fichier suivant (`JournalBlock.java`), qui représente un bloc de journal et son contenu de données. Il inclut une méthode nommée`verifyBlockHash()` qui montre comment calculer chaque composant individuel d'un hachage de bloc.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.qldb;

import com.fasterxml.jackson.annotation.JsonCreator;
```

```
import com.fasterxml.jackson.annotation.JsonProperty;
import com.fasterxml.jackson.databind.annotation.JsonSerialize;
import com.fasterxml.jackson.dataformat.ion.IonTimestampSerializers;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.qldb.tutorial.Constants;
import software.amazon.qldb.tutorial.Verifier;

import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Arrays;
import java.util.Date;
import java.util.HashSet;
import java.util.List;
import java.util.Set;
import java.util.stream.Collectors;

import static java.nio.ByteBuffer.wrap;

/**
 * Represents a JournalBlock that was recorded after executing a transaction
 * in the ledger.
 */
public final class JournalBlock {
    private static final Logger log = LoggerFactory.getLogger(JournalBlock.class);

    private BlockAddress blockAddress;
    private String transactionId;
    @JsonSerialize(using =
 IonTimestampSerializers.IonTimestampJavaDateSerializer.class)
    private Date blockTimestamp;
    private byte[] blockHash;
    private byte[] entriesHash;
    private byte[] previousBlockHash;
    private byte[][] entriesHashList;
    private TransactionInfo transactionInfo;
    private RedactionInfo redactionInfo;
    private List<QldbRevision> revisions;

    @JsonCreator
    public JournalBlock(@JsonProperty("blockAddress") final BlockAddress
 blockAddress,
                        @JsonProperty("transactionId") final String transactionId,
                        @JsonProperty("blockTimestamp") final Date blockTimestamp,
```

```
                        @JsonProperty("blockHash") final byte[] blockHash,
                        @JsonProperty("entriesHash") final byte[] entriesHash,
                        @JsonProperty("previousBlockHash") final byte[]
previousBlockHash,
                        @JsonProperty("entriesHashList") final byte[][]
entriesHashList,
                        @JsonProperty("transactionInfo") final TransactionInfo
transactionInfo,
                        @JsonProperty("redactionInfo") final RedactionInfo
redactionInfo,
                        @JsonProperty("revisions") final List<QldbRevision>
revisions) {
       this.blockAddress = blockAddress;
       this.transactionId = transactionId;
       this.blockTimestamp = blockTimestamp;
       this.blockHash = blockHash;
       this.entriesHash = entriesHash;
       this.previousBlockHash = previousBlockHash;
       this.entriesHashList = entriesHashList;
       this.transactionInfo = transactionInfo;
       this.redactionInfo = redactionInfo;
       this.revisions = revisions;
   }

   public BlockAddress getBlockAddress() {
       return blockAddress;
   }

   public String getTransactionId() {
       return transactionId;
   }

   public Date getBlockTimestamp() {
       return blockTimestamp;
   }

   public byte[][] getEntriesHashList() {
       return entriesHashList;
   }

   public TransactionInfo getTransactionInfo() {
       return transactionInfo;
   }
```

```
    public RedactionInfo getRedactionInfo() {
        return redactionInfo;
    }

    public List<QldbRevision> getRevisions() {
        return revisions;
    }

    public byte[] getEntriesHash() {
        return entriesHash;
    }

    public byte[] getBlockHash() {
        return blockHash;
    }

    public byte[] getPreviousBlockHash() {
        return previousBlockHash;
    }

    @Override
    public String toString() {
        return "JournalBlock{"
                + "blockAddress=" + blockAddress
                + ", transactionId='" + transactionId + '\''
                + ", blockTimestamp=" + blockTimestamp
                + ", blockHash=" + Arrays.toString(blockHash)
                + ", entriesHash=" + Arrays.toString(entriesHash)
                + ", previousBlockHash=" + Arrays.toString(previousBlockHash)
                + ", entriesHashList=" + Arrays.toString(entriesHashList)
                + ", transactionInfo=" + transactionInfo
                + ", redactionInfo=" + redactionInfo
                + ", revisions=" + revisions
                + '}';
    }

    @Override
    public boolean equals(final Object o) {
        if (this == o) {
            return true;
        }
        if (!(o instanceof JournalBlock)) {
            return false;
        }
```

```
        final JournalBlock that = (JournalBlock) o;

        if (!getBlockAddress().equals(that.getBlockAddress())) {
            return false;
        }
        if (!getTransactionId().equals(that.getTransactionId())) {
            return false;
        }
        if (!getBlockTimestamp().equals(that.getBlockTimestamp())) {
            return false;
        }
        if (!Arrays.equals(getBlockHash(), that.getBlockHash())) {
            return false;
        }
        if (!Arrays.equals(getEntriesHash(), that.getEntriesHash())) {
            return false;
        }
        if (!Arrays.equals(getPreviousBlockHash(), that.getPreviousBlockHash())) {
            return false;
        }
        if (!Arrays.deepEquals(getEntriesHashList(), that.getEntriesHashList())) {
            return false;
        }
        if (!getTransactionInfo().equals(that.getTransactionInfo())) {
            return false;
        }
        if (getRedactionInfo() != null ? !
getRedactionInfo().equals(that.getRedactionInfo()) : that.getRedactionInfo() !=
 null) {
            return false;
        }
        return getRevisions() != null ?
 getRevisions().equals(that.getRevisions()) : that.getRevisions() == null;
    }

    @Override
    public int hashCode() {
        int result = getBlockAddress().hashCode();
        result = 31 * result + getTransactionId().hashCode();
        result = 31 * result + getBlockTimestamp().hashCode();
        result = 31 * result + Arrays.hashCode(getBlockHash());
        result = 31 * result + Arrays.hashCode(getEntriesHash());
        result = 31 * result + Arrays.hashCode(getPreviousBlockHash());
```

```
        result = 31 * result + Arrays.deepHashCode(getEntriesHashList());
        result = 31 * result + getTransactionInfo().hashCode();
        result = 31 * result + (getRedactionInfo() != null ?
getRedactionInfo().hashCode() : 0);
        result = 31 * result + (getRevisions() != null ?
getRevisions().hashCode() : 0);
        return result;
    }

    /**
     * This method validates that the hashes of the components of a journal block
make up the block
     * hash that is provided with the block itself.
     *
     * The components that contribute to the hash of the journal block consist of
the following:
     *   - user transaction information (contained in [transactionInfo])
     *   - user redaction information (contained in [redactionInfo])
     *   - user revisions (contained in [revisions])
     *   - hashes of internal-only system metadata (contained in [revisions] and in
[entriesHashList])
     *   - the previous block hash
     *
     * If any of the computed hashes of user information cannot be validated or any
of the system
     * hashes do not result in the correct computed values, this method will throw
an IllegalArgumentException.
     *
     * Internal-only system metadata is represented by its hash, and can be present
in the form of certain
     * items in the [revisions] list that only contain a hash and no user data, as
well as some hashes
     * in [entriesHashList].
     *
     * To validate that the hashes of the user data are valid components of the
[blockHash], this method
     * performs the following steps:
     *
     * 1. Compute the hash of the [transactionInfo] and validate that it is
included in the [entriesHashList].
     * 2. Compute the hash of the [redactionInfo], if present, and validate that it
is included in the [entriesHashList].
     * 3. Validate the hash of each user revision was correctly computed and
matches the hash published
```

```
     * with that revision.
     * 4. Compute the hash of the [revisions] by treating the revision hashes as
 the leaf nodes of a Merkle tree
     * and calculating the root hash of that tree. Then validate that hash is
 included in the [entriesHashList].
     * 5. Compute the hash of the [entriesHashList] by treating the hashes as the
 leaf nodes of a Merkle tree
     * and calculating the root hash of that tree. Then validate that hash matches
 [entriesHash].
     * 6. Finally, compute the block hash by computing the hash resulting from
 concatenating the [entriesHash]
     * and previous block hash, and validate that the result matches the
 [blockHash] provided by QLDB with the block.
     *
     * This method is called by ValidateQldbHashChain::verify for each journal
 block to validate its
     * contents before verifying that the hash chain between consecutive blocks is
 correct.
     */
    public void verifyBlockHash() {
        Set<ByteBuffer> entriesHashSet = new HashSet<>();
        Arrays.stream(entriesHashList).forEach(hash ->
 entriesHashSet.add(wrap(hash).asReadOnlyBuffer()));

        byte[] computedTransactionInfoHash = computeTransactionInfoHash();
        if (!
entriesHashSet.contains(wrap(computedTransactionInfoHash).asReadOnlyBuffer())) {
            throw new IllegalArgumentException(
                    "Block transactionInfo hash is not contained in the QLDB block
 entries hash list.");
        }

        if (redactionInfo != null) {
            byte[] computedRedactionInfoHash = computeRedactionInfoHash();
            if (!
entriesHashSet.contains(wrap(computedRedactionInfoHash).asReadOnlyBuffer())) {
                throw new IllegalArgumentException(
                        "Block redactionInfo hash is not contained in the QLDB
 block entries hash list.");
            }
        }

        if (revisions != null) {
            revisions.forEach(QldbRevision::verifyRevisionHash);
```

```
            byte[] computedRevisionsHash = computeRevisionsHash();
            if (!
entriesHashSet.contains(wrap(computedRevisionsHash).asReadOnlyBuffer())) {
                throw new IllegalArgumentException(
                        "Block revisions list hash is not contained in the QLDB
 block entries hash list.");
            }
        }

        byte[] computedEntriesHash = computeEntriesHash();
        if (!Arrays.equals(computedEntriesHash, entriesHash)) {
            throw new IllegalArgumentException("Computed entries hash does not
 match entries hash provided in the block.");
        }

        byte[] computedBlockHash = Verifier.dot(computedEntriesHash,
 previousBlockHash);
        if (!Arrays.equals(computedBlockHash, blockHash)) {
            throw new IllegalArgumentException("Computed block hash does not match
 block hash provided in the block.");
        }
    }

    private byte[] computeTransactionInfoHash() {
        try {
            return
 QldbIonUtils.hashIonValue(Constants.MAPPER.writeValueAsIonValue(transactionInfo));
        } catch (IOException e) {
            throw new IllegalArgumentException("Could not compute transactionInfo
 hash to verify block hash.", e);
        }
    }

    private byte[] computeRedactionInfoHash() {
        try {
            return
 QldbIonUtils.hashIonValue(Constants.MAPPER.writeValueAsIonValue(redactionInfo));
        } catch (IOException e) {
            throw new IllegalArgumentException("Could not compute redactionInfo
 hash to verify block hash.", e);
        }
    }

    private byte[] computeRevisionsHash() {
```

```
        return
 Verifier.calculateMerkleTreeRootHash(revisions.stream().map(QldbRevision::getHash).collect
    }

    private byte[] computeEntriesHash() {
        return
 Verifier.calculateMerkleTreeRootHash(Arrays.asList(entriesHashList));
    }
}
```

2. Compilez et exécutez le programme suivant (`ValidateQldbHashChain.java`) pour effectuer les étapes suivantes :

   1. Exportez des blocs de journal depuis le`vehicle-registration` registre vers un compartiment Amazon S3 nommé**`qldb-tutorial-journal-export-`*`111122223333`*** (remplacez-le par votre`Compte AWS` numéro).
   2. Validez les composants de hachage individuels au sein de chaque bloc en appelant`verifyBlockHash()`.
   3. Validez la chaîne de hachage entre les blocs de journal.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
```

```
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazonaws.services.qldb.model.ExportJournalToS3Result;
import com.amazonaws.services.qldb.model.S3EncryptionConfiguration;
import com.amazonaws.services.qldb.model.S3ObjectEncryptionType;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;

import java.time.Instant;
import java.util.Arrays;
import java.util.List;

import com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClientBuilder;
import com.amazonaws.services.securitytoken.model.GetCallerIdentityRequest;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import software.amazon.qldb.tutorial.qldb.JournalBlock;

/**
 * Validate the hash chain of a QLDB ledger by stepping through its S3 export.
 *
 * This code accepts an exportId as an argument, if exportId is passed the code
 * will use that or request QLDB to generate a new export to perform QLDB hash
 * chain validation.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class ValidateQldbHashChain {
    public static final Logger log =
 LoggerFactory.getLogger(ValidateQldbHashChain.class);
    private static final int TIME_SKEW = 20;

    private ValidateQldbHashChain() { }

    /**
     * Export journal contents to a S3 bucket.
     *
     * @return the ExportId of the journal export.
```

```
    * @throws InterruptedException if the thread is interrupted while waiting for
export to complete.
    */
   private static String createExport() throws InterruptedException {
       String accountId = AWSSecurityTokenServiceClientBuilder.defaultClient()
           .getCallerIdentity(new GetCallerIdentityRequest()).getAccount();
       String bucketName = Constants.JOURNAL_EXPORT_S3_BUCKET_NAME_PREFIX + "-" +
accountId;
       String prefix = Constants.LEDGER_NAME + "-" +
Instant.now().getEpochSecond() + "/";

       S3EncryptionConfiguration encryptionConfiguration = new
S3EncryptionConfiguration()
               .withObjectEncryptionType(S3ObjectEncryptionType.SSE_S3);
       ExportJournalToS3Result exportJournalToS3Result =

ExportJournal.createJournalExportAndAwaitCompletion(Constants.LEDGER_NAME,
                   bucketName, prefix, null, encryptionConfiguration,
ExportJournal.DEFAULT_EXPORT_TIMEOUT_MS);

       return exportJournalToS3Result.getExportId();
   }

   /**
    * Validates that the chain hash on the {@link JournalBlock} is valid.
    *
    * @param journalBlocks
    *              {@link JournalBlock} containing hashes to validate.
    * @throws IllegalStateException if previous block hash does not match.
    */
   public static void verify(final List<JournalBlock> journalBlocks) {
       if (journalBlocks.size() == 0) {
           return;
       }

       journalBlocks.stream().reduce(null, (previousJournalBlock, journalBlock) ->
{
           journalBlock.verifyBlockHash();
           if (previousJournalBlock == null) { return journalBlock; }
           if (!Arrays.equals(previousJournalBlock.getBlockHash(),
journalBlock.getPreviousBlockHash())) {
               throw new IllegalStateException("Previous block hash doesn't
match.");
           }
```

```
            byte[] blockHash = Verifier.dot(journalBlock.getEntriesHash(),
 previousJournalBlock.getBlockHash());
            if (!Arrays.equals(blockHash, journalBlock.getBlockHash())) {
                throw new IllegalStateException("Block hash doesn't match
 entriesHash dot previousBlockHash, the chain is "
                        + "broken.");
            }
            return journalBlock;
        });
    }

    public static void main(final String... args) throws InterruptedException {
        try {
            String exportId;
            if (args.length == 1) {
                exportId = args[0];
                log.info("Validating QLDB hash chain for exportId: " + exportId);
            } else {
                log.info("Requesting QLDB to create an export.");
                exportId = createExport();
            }
            List<JournalBlock> journalBlocks =

 JournalS3ExportReader.readExport(DescribeJournalExport.describeExport(Constants.LEDGER_NAM
                    exportId), AmazonS3ClientBuilder.defaultClient());
            verify(journalBlocks);
        } catch (Exception e) {
            log.error("Unable to perform hash chain verification.", e);
            throw e;
        }
    }

}
```

Si vous n'avez plus besoin d'utiliser le`vehicle-registration` registre, passez àÉtape 9 (facultatif) : Nettoyer les ressources.

Étape 9 (facultatif) : Nettoyer les ressources

Vous pouvez continuer à utiliser le`vehicle-registration` registre. Toutefois, si vous n'en avez plus besoin, vous devez le supprimer.

Pour supprimer le registre

1. Compilez et exécutez le programme suivant (`DeleteLedger.java`) pour supprimer votre`vehicle-registration` registre et tout son contenu.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazonaws.services.qldb.AmazonQLDB;
import com.amazonaws.services.qldb.model.DeleteLedgerRequest;
import com.amazonaws.services.qldb.model.DeleteLedgerResult;
import com.amazonaws.services.qldb.model.ResourceNotFoundException;
import com.amazonaws.services.qldb.model.UpdateLedgerRequest;
import com.amazonaws.services.qldb.model.UpdateLedgerResult;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
```

```
 * Delete a ledger.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class DeleteLedger {
    public static final Logger log = LoggerFactory.getLogger(DeleteLedger.class);
    public static final Long LEDGER_DELETION_POLL_PERIOD_MS = 20_000L;
    public static AmazonQLDB client = CreateLedger.getClient();

    private DeleteLedger() { }

    public static void main(String... args) throws Exception {
        try {
            setDeletionProtection(Constants.LEDGER_NAME, false);

            delete(Constants.LEDGER_NAME);

            waitForDeleted(Constants.LEDGER_NAME);

        } catch (Exception e) {
            log.error("Unable to delete the ledger.", e);
            throw e;
        }
    }

    /**
     * Send a request to the QLDB database to delete the specified ledger.
     *
     * @param ledgerName
     *              Name of the ledger to be deleted.
     * @return DeleteLedgerResult.
     */
    public static DeleteLedgerResult delete(final String ledgerName) {
        log.info("Attempting to delete the ledger with name: {}...", ledgerName);
        DeleteLedgerRequest request = new
 DeleteLedgerRequest().withName(ledgerName);
        DeleteLedgerResult result = client.deleteLedger(request);
        log.info("Success.");
        return result;
    }

    /**
```

```
     * Wait for the ledger to be deleted.
     *
     * @param ledgerName
     *              Name of the ledger being deleted.
     * @throws InterruptedException if thread is being interrupted.
     */
    public static void waitForDeleted(final String ledgerName) throws
 InterruptedException {
        log.info("Waiting for the ledger to be deleted...");
        while (true) {
            try {
                DescribeLedger.describe(ledgerName);
                log.info("The ledger is still being deleted. Please wait...");
                Thread.sleep(LEDGER_DELETION_POLL_PERIOD_MS);
            } catch (ResourceNotFoundException ex) {
                log.info("Success. The ledger is deleted.");
                break;
            }
        }
    }

    public static UpdateLedgerResult setDeletionProtection(String ledgerName,
 boolean deletionProtection) {
        log.info("Let's set deletionProtection to {} for the ledger with name {}",
 deletionProtection, ledgerName);
        UpdateLedgerRequest request = new UpdateLedgerRequest()
                .withName(ledgerName)
                .withDeletionProtection(deletionProtection);

        UpdateLedgerResult result = client.updateLedger(request);
        log.info("Success. Ledger updated: {}", result);
        return result;
    }
}
```

> **Note**
>
> Si la protection contre la suppression est activée pour votre registre, vous devez commencer par la désactiver avant de pouvoir supprimer le registre à l'aide de l'API QLDB.

2. Si vous avez exporté les données du journal à l'étape précédente et que vous n'en avez plus besoin, utilisez la console Amazon S3 pour supprimer votre compartiment S3.

   Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.

Tutoriel Amazon QLDB Node.js

Dans cette implémentation de l'exemple d'application du didacticiel, vous utilisez le pilote Amazon QLDB avec AWS le SDK JavaScript pour Node.js afin de créer un registre QLDB et de le remplir avec des exemples de données.

Au cours de ce didacticiel, vous pouvez vous référer à la référence d'AWS SDK for JavaScriptAPI pour les opérations d'API de gestion. Pour les opérations relatives aux données transactionnelles, vous pouvez consulter le guide de référence de l'API QLDB pour Node.js.

> **Note**
>
> Le cas échéant, certaines étapes du didacticiel proposent des commandes ou des exemples de code différents pour chaque version majeure prise en charge du pilote QLDB pour Node.js.

Rubriques

Installation de l'exemple d'application Amazon QLDB Node.js

Cette section décrit comment installer et exécuter l'exemple d'application Amazon QLDB fourni pour step-by-step le didacticiel Node.js. Le cas d'utilisation de cet exemple d'application est une base de données du département des véhicules automobiles (DMV) qui permet de suivre les informations historiques complètes sur les immatriculations de véhicules.

L'exemple d'application DMV pour Node.js est open source dans le GitHub référentiel amazon-qldb-dmv-sampleaws-samples/ -nodejs.

Prérequis

Avant de commencer, assurez-vous d'avoir terminé le pilote QLDB pour Node.js. Prérequis Cela inclut l'installation de Node.js et les opérations suivantes :

1. Inscrivez-vous pourAWS.
2. Créez un utilisateur doté des autorisations QLDB appropriées.
3. Accordez un accès programmatique pour le développement.

Pour effectuer toutes les étapes de ce didacticiel, vous devez disposer d'un accès administratif complet à votre ressource de registre via l'API QLDB.

Installation

Pour installer l'exemple d'application

1. Entrez la commande suivante pour cloner l'exemple d'application GitHub.

   2.x

   ```
   git clone https://github.com/aws-samples/amazon-qldb-dmv-sample-nodejs.git
   ```

   1.x

   ```
   git clone -b v1.0.0 https://github.com/aws-samples/amazon-qldb-dmv-sample-
   nodejs.git
   ```

L'exemple d'application contient le code source complet de ce didacticiel et ses dépendances, y compris le pilote Node.js et le AWSSDK pour JavaScript Node.js. Cette application est écrite en TypeScript.

2. Accédez au répertoire dans lequel le `amazon-qldb-dmv-sample-nodejs` package est cloné.

```
cd amazon-qldb-dmv-sample-nodejs
```

3. Procédez à une nouvelle installation des dépendances.

```
npm ci
```

4. Transpilez le colis.

```
npm run build
```

Les JavaScript fichiers transpilés sont écrits dans le `./dist` répertoire.

5. Procédez Étape 1 : Création d'un nouveau registre à pour démarrer le didacticiel et créer un registre.

Étape 1 : Création d'un nouveau registre

Au cours de cette étape, vous allez créer un nouveau registre Amazon QLDB nommé. `vehicle-registration`

Pour créer un nouveau registre

1. Consultez le fichier suivant (`Constants.ts`), qui contient les valeurs constantes utilisées par tous les autres programmes de ce didacticiel.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
```

```
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

/**
 * Constant values used throughout this tutorial.
 */
export const LEDGER_NAME = "vehicle-registration";
export const LEDGER_NAME_WITH_TAGS = "tags";

export const DRIVERS_LICENSE_TABLE_NAME = "DriversLicense";
export const PERSON_TABLE_NAME = "Person";
export const VEHICLE_REGISTRATION_TABLE_NAME = "VehicleRegistration";
export const VEHICLE_TABLE_NAME = "Vehicle";

export const GOV_ID_INDEX_NAME = "GovId";
export const LICENSE_NUMBER_INDEX_NAME = "LicenseNumber";
export const LICENSE_PLATE_NUMBER_INDEX_NAME = "LicensePlateNumber";
export const PERSON_ID_INDEX_NAME = "PersonId";
export const VIN_INDEX_NAME = "VIN";

export const RETRY_LIMIT = 4;

export const JOURNAL_EXPORT_S3_BUCKET_NAME_PREFIX = "qldb-tutorial-journal-export";
export const USER_TABLES = "information_schema.user_tables";
```

2. Utilisez le programme suivant (`CreateLedger.ts`) pour créer un registre nommé`vehicle-registration`.

```
/*
```

```
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QLDB } from "aws-sdk";
import {
    CreateLedgerRequest,
    CreateLedgerResponse,
    DescribeLedgerRequest,
    DescribeLedgerResponse
} from "aws-sdk/clients/qldb";

import { LEDGER_NAME } from "./qldb/Constants";
import { error, log } from "./qldb/LogUtil";
import { sleep } from "./qldb/Util";

const LEDGER_CREATION_POLL_PERIOD_MS = 10000;
const ACTIVE_STATE = "ACTIVE";

/**
 * Create a new ledger with the specified name.
 * @param ledgerName Name of the ledger to be created.
 * @param qldbClient The QLDB control plane client to use.
 * @returns Promise which fulfills with a CreateLedgerResponse.
```

```
 */
export async function createLedger(ledgerName: string, qldbClient: QLDB):
 Promise<CreateLedgerResponse> {
    log(`Creating a ledger named: ${ledgerName}...`);
    const request: CreateLedgerRequest = {
        Name: ledgerName,
        PermissionsMode: "ALLOW_ALL"
    }
    const result: CreateLedgerResponse = await
 qldbClient.createLedger(request).promise();
    log(`Success. Ledger state: ${result.State}.`);
    return result;
}

/**
 * Wait for the newly created ledger to become active.
 * @param ledgerName Name of the ledger to be checked on.
 * @param qldbClient The QLDB control plane client to use.
 * @returns Promise which fulfills with a DescribeLedgerResponse.
 */
export async function waitForActive(ledgerName: string, qldbClient: QLDB):
 Promise<DescribeLedgerResponse> {
    log(`Waiting for ledger ${ledgerName} to become active...`);
    const request: DescribeLedgerRequest = {
        Name: ledgerName
    }
    while (true) {
        const result: DescribeLedgerResponse = await
 qldbClient.describeLedger(request).promise();
        if (result.State === ACTIVE_STATE) {
            log("Success. Ledger is active and ready to be used.");
            return result;
        }
        log("The ledger is still creating. Please wait...");
        await sleep(LEDGER_CREATION_POLL_PERIOD_MS);
    }
}

/**
 * Create a ledger and wait for it to be active.
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
    try {
```

```
        const qldbClient: QLDB = new QLDB();
        await createLedger(LEDGER_NAME, qldbClient);
        await waitForActive(LEDGER_NAME, qldbClient);
    } catch (e) {
        error(`Unable to create the ledger: ${e}`);
    }
}

if (require.main === module) {
    main();
}
```

> **Note**
>
> - Lors de l'`createLedger`appel, vous devez spécifier un nom de registre et un mode d'autorisation. Nous vous recommandons d'utiliser le mode `STANDARD` autorisations pour optimiser la sécurité des données de votre registre.
> - Lorsque vous créez un registre, la protection contre les suppressions est activée par défaut. Il s'agit d'une fonctionnalité de QLDB qui empêche la suppression de registres par un utilisateur. Vous avez la possibilité de désactiver la protection contre la suppression lors de la création du registre à l'aide de l'API QLDB ou du (). AWS Command Line Interface AWS CLI
> - En option, vous pouvez également spécifier des balises à associer à votre registre.

3. Pour exécuter le programme transpilé, entrez la commande suivante.

```
node dist/CreateLedger.js
```

Pour vérifier votre connexion au nouveau registre, passez àÉtape 2 : tester la connectivité au registre.

Étape 2 : tester la connectivité au registre

Au cours de cette étape, vous devez vérifier que vous pouvez vous connecter au `vehicle-registration` registre dans Amazon QLDB à l'aide du point de terminaison de l'API de données transactionnelles.

Pour tester la connectivité au registre

1. Utilisez le programme suivant (`ConnectToLedger.ts`) pour créer une connexion de session de données au `vehicle-registration` registre.

   2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QldbDriver, RetryConfig  } from "amazon-qldb-driver-nodejs";
import { ClientConfiguration } from "aws-sdk/clients/qldbsession";

import { LEDGER_NAME } from "./qldb/Constants";
import { error, log } from "./qldb/LogUtil";

const qldbDriver: QldbDriver = createQldbDriver();

/**
 * Create a driver for creating sessions.
```

```
 * @param ledgerName The name of the ledger to create the driver on.
 * @param serviceConfigurationOptions The configurations for the AWS SDK client
 that the driver uses.
 * @returns The driver for creating sessions.
 */
export function createQldbDriver(
    ledgerName: string = LEDGER_NAME,
    serviceConfigurationOptions: ClientConfiguration = {}
): QldbDriver {
    const retryLimit = 4;
    const maxConcurrentTransactions = 10;
    //Use driver's default backoff function (and hence, no second parameter
 provided to RetryConfig)
    const retryConfig: RetryConfig = new RetryConfig(retryLimit);
    const qldbDriver: QldbDriver = new QldbDriver(ledgerName,
 serviceConfigurationOptions, maxConcurrentTransactions, retryConfig);
    return qldbDriver;
}

export function getQldbDriver(): QldbDriver {
    return qldbDriver;
}

/**
 * Connect to a session for a given ledger using default settings.
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
    try {
        log("Listing table names...");
        const tableNames: string[] = await qldbDriver.getTableNames();
        tableNames.forEach((tableName: string): void => {
            log(tableName);
        });
    } catch (e) {
        error(`Unable to create session: ${e}`);
    }
}

if (require.main === module) {
    main();
}
```

1.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QldbDriver  } from "amazon-qldb-driver-nodejs";
import { ClientConfiguration } from "aws-sdk/clients/qldbsession";

import { LEDGER_NAME } from "./qldb/Constants";
import { error, log } from "./qldb/LogUtil";

const qldbDriver: QldbDriver = createQldbDriver();

/**
 * Create a driver for creating sessions.
 * @param ledgerName The name of the ledger to create the driver on.
 * @param serviceConfigurationOptions The configurations for the AWS SDK client
 that the driver uses.
 * @returns The driver for creating sessions.
 */
```

```
export function createQldbDriver(
    ledgerName: string = LEDGER_NAME,
    serviceConfigurationOptions: ClientConfiguration = {}
): QldbDriver {
    const qldbDriver: QldbDriver = new QldbDriver(ledgerName,
 serviceConfigurationOptions);
    return qldbDriver;
}

export function getQldbDriver(): QldbDriver {
    return qldbDriver;
}

/**
 * Connect to a session for a given ledger using default settings.
 * @returns Promise which fulfills with void.
 */
var main = async function(): Promise<void> {
    try {
        log("Listing table names...");
        const tableNames: string[] = await qldbDriver.getTableNames();
        tableNames.forEach((tableName: string): void => {
            log(tableName);
        });
    } catch (e) {
        error(`Unable to create session: ${e}`);
    }
}

if (require.main === module) {
    main();
}
```

> **Note**
>
> Pour exécuter des transactions de données sur votre registre, vous devez créer un objet pilote QLDB pour vous connecter à un registre spécifié. Il s'agit d'un objet client différent de celui que vous avez utilisé à l'étape précédente pour créer le registre. `qldbClient` Ce client précédent est uniquement utilisé pour exécuter les opérations d'API de gestion répertoriées dans leRéférence d'API Amazon QLDB.

2. Pour exécuter le programme transpilé, entrez la commande suivante.

```
node dist/ConnectToLedger.js
```

Pour créer des tables dans le `vehicle-registration` registre, passez àÉtape 3 : créer des tables, des index et des exemples de données.

Étape 3 : créer des tables, des index et des exemples de données

Lorsque votre registre Amazon QLDB est actif et accepte les connexions, vous pouvez commencer à créer des tableaux contenant les données relatives aux véhicules, à leurs propriétaires et à leurs informations d'enregistrement. Après avoir créé les tables et les index, vous pouvez les charger avec des données.

Au cours de cette étape, vous créez quatre tables dans le `vehicle-registration` registre :

- `VehicleRegistration`
- `Vehicle`
- `Person`
- `DriversLicense`

Vous créez également les index suivants.

Nom de la table	Champ
`VehicleRegistration`	`VIN`
`VehicleRegistration`	`LicensePlateNumber`
`Vehicle`	`VIN`
`Person`	`GovId`
`DriversLicense`	`LicenseNumber`
`DriversLicense`	`PersonId`

Lorsque vous insérez des exemples de données, vous devez d'abord insérer des documents dans le `Person` tableau. Ensuite, vous utilisez le système attribué `id` à chaque `Person` document pour remplir les champs correspondants dans les documents appropriés`VehicleRegistration`. `DriversLicense`

> **Tip**
>
> La meilleure pratique consiste à utiliser le système attribué à un document `id` en tant que clé étrangère. Bien que vous puissiez définir des champs destinés à être des identifiants uniques (par exemple, le VIN d'un véhicule), le véritable identifiant unique d'un document est le sien`id`. Ce champ est inclus dans les métadonnées du document, que vous pouvez interroger dans la vue validée (la vue définie par le système d'une table).
> Pour plus d'informations sur les vues dans QLDB, consultez. Concepts de base Pour en savoir plus sur les métadonnées, consultezInterroger les métadonnées d'un document.

Pour créer des tables et des index

1. Utilisez le programme suivant (`CreateTable.ts`) pour créer les tables mentionnées précédemment.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
```

```
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QldbDriver, Result, TransactionExecutor } from "amazon-qldb-driver-
nodejs";

import { getQldbDriver } from "./ConnectToLedger";
import {
    DRIVERS_LICENSE_TABLE_NAME,
    PERSON_TABLE_NAME,
    VEHICLE_REGISTRATION_TABLE_NAME,
    VEHICLE_TABLE_NAME
} from "./qldb/Constants";
import { error, log } from "./qldb/LogUtil";

/**
 * Create multiple tables in a single transaction.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param tableName Name of the table to create.
 * @returns Promise which fulfills with the number of changes to the database.
 */
export async function createTable(txn: TransactionExecutor, tableName: string):
 Promise<number> {
    const statement: string = `CREATE TABLE ${tableName}`;
    return await txn.execute(statement).then((result: Result) => {
        log(`Successfully created table ${tableName}.`);
        return result.getResultList().length;
    });
}

/**
 * Create tables in a QLDB ledger.
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
    try {
        const qldbDriver: QldbDriver = getQldbDriver();
        await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
            Promise.all([
                createTable(txn, VEHICLE_REGISTRATION_TABLE_NAME),
                createTable(txn, VEHICLE_TABLE_NAME),
```

```
                createTable(txn, PERSON_TABLE_NAME),
                createTable(txn, DRIVERS_LICENSE_TABLE_NAME)
            ]);
        });
    } catch (e) {
        error(`Unable to create tables: ${e}`);
    }
}

if (require.main === module) {
    main();
}
```

> **Note**
>
> Ce programme explique comment utiliser la `executeLambda` fonction dans une instance de pilote QLDB. Dans cet exemple, vous exécutez plusieurs instructions `CREATE TABLE` PartiQL avec une seule expression lambda.
> Cette fonction d'exécution démarre implicitement une transaction, exécute toutes les instructions du lambda, puis valide automatiquement la transaction.

2. Pour exécuter le programme transpilé, entrez la commande suivante.

```
node dist/CreateTable.js
```

3. Utilisez le programme suivant (`CreateIndex.ts`) pour créer des index sur les tables, comme décrit précédemment.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
```

```
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QldbDriver, TransactionExecutor } from "amazon-qldb-driver-nodejs";

import { getQldbDriver } from "./ConnectToLedger";
import {
    DRIVERS_LICENSE_TABLE_NAME,
    GOV_ID_INDEX_NAME,
    LICENSE_NUMBER_INDEX_NAME,
    LICENSE_PLATE_NUMBER_INDEX_NAME,
    PERSON_ID_INDEX_NAME,
    PERSON_TABLE_NAME,
    VEHICLE_REGISTRATION_TABLE_NAME,
    VEHICLE_TABLE_NAME,
    VIN_INDEX_NAME
} from "./qldb/Constants";
import { error, log } from "./qldb/LogUtil";

/**
 * Create an index for a particular table.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param tableName Name of the table to add indexes for.
 * @param indexAttribute Index to create on a single attribute.
 * @returns Promise which fulfills with the number of changes to the database.
 */
export async function createIndex(
    txn: TransactionExecutor,
    tableName: string,
    indexAttribute: string
): Promise<number> {
    const statement: string = `CREATE INDEX on ${tableName} (${indexAttribute})`;
    return await txn.execute(statement).then((result) => {
        log(`Successfully created index ${indexAttribute} on table ${tableName}.`);
        return result.getResultList().length;
```

```
    });
}

/**
 * Create indexes on tables in a particular ledger.
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
    try {
        const qldbDriver: QldbDriver = getQldbDriver();
        await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
            Promise.all([
                createIndex(txn, PERSON_TABLE_NAME, GOV_ID_INDEX_NAME),
                createIndex(txn, VEHICLE_TABLE_NAME, VIN_INDEX_NAME),
                createIndex(txn, VEHICLE_REGISTRATION_TABLE_NAME, VIN_INDEX_NAME),
                createIndex(txn, VEHICLE_REGISTRATION_TABLE_NAME,
 LICENSE_PLATE_NUMBER_INDEX_NAME),
                createIndex(txn, DRIVERS_LICENSE_TABLE_NAME, PERSON_ID_INDEX_NAME),
                createIndex(txn, DRIVERS_LICENSE_TABLE_NAME,
 LICENSE_NUMBER_INDEX_NAME)
            ]);
        });
    } catch (e) {
        error(`Unable to create indexes: ${e}`);
    }
}

if (require.main === module) {
    main();
}
```

4. Pour exécuter le programme transpilé, entrez la commande suivante.

```
node dist/CreateIndex.js
```

Pour charger des données dans les tables

1. Passez en revue les `.ts` fichiers suivants.

    1. `SampleData.ts`— Contient les exemples de données que vous insérez dans les `vehicle-registration` tables.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { Decimal } from "ion-js";

const EMPTY_SECONDARY_OWNERS: object[] = [];
export const DRIVERS_LICENSE = [
    {
        PersonId: "",
        LicenseNumber: "LEWISR261LL",
        LicenseType: "Learner",
        ValidFromDate: new Date("2016-12-20"),
        ValidToDate: new Date("2020-11-15")
    },
    {
        PersonId: "",
        LicenseNumber : "LOGANB486CG",
        LicenseType: "Probationary",
        ValidFromDate : new Date("2016-04-06"),
        ValidToDate : new Date("2020-11-15")
    },
```

```
    {
        PersonId: "",
        LicenseNumber : "744 849 301",
        LicenseType: "Full",
        ValidFromDate : new Date("2017-12-06"),
        ValidToDate : new Date("2022-10-15")
    },
    {
        PersonId: "",
        LicenseNumber : "P626-168-229-765",
        LicenseType: "Learner",
        ValidFromDate : new Date("2017-08-16"),
        ValidToDate : new Date("2021-11-15")
    },
    {
        PersonId: "",
        LicenseNumber : "S152-780-97-415-0",
        LicenseType: "Probationary",
        ValidFromDate : new Date("2015-08-15"),
        ValidToDate : new Date("2021-08-21")
    }
];
export const PERSON = [
    {
        FirstName : "Raul",
        LastName : "Lewis",
        DOB : new Date("1963-08-19"),
        Address : "1719 University Street, Seattle, WA, 98109",
        GovId : "LEWISR261LL",
        GovIdType : "Driver License"
    },
    {
        FirstName : "Brent",
        LastName : "Logan",
        DOB : new Date("1967-07-03"),
        Address : "43 Stockert Hollow Road, Everett, WA, 98203",
        GovId : "LOGANB486CG",
        GovIdType : "Driver License"
    },
    {
        FirstName : "Alexis",
        LastName : "Pena",
        DOB : new Date("1974-02-10"),
        Address : "4058 Melrose Street, Spokane Valley, WA, 99206",
```

```
        GovId : "744 849 301",
        GovIdType : "SSN"
    },
    {
        FirstName : "Melvin",
        LastName : "Parker",
        DOB : new Date("1976-05-22"),
        Address : "4362 Ryder Avenue, Seattle, WA, 98101",
        GovId : "P626-168-229-765",
        GovIdType : "Passport"
    },
    {
        FirstName : "Salvatore",
        LastName : "Spencer",
        DOB : new Date("1997-11-15"),
        Address : "4450 Honeysuckle Lane, Seattle, WA, 98101",
        GovId : "S152-780-97-415-0",
        GovIdType : "Passport"
    }
];
export const VEHICLE = [
    {
        VIN : "1N4AL11D75C109151",
        Type : "Sedan",
        Year : 2011,
        Make : "Audi",
        Model : "A5",
        Color : "Silver"
    },
    {
        VIN : "KM8SRDHF6EU074761",
        Type : "Sedan",
        Year : 2015,
        Make : "Tesla",
        Model : "Model S",
        Color : "Blue"
    },
    {
        VIN : "3HGGK5G53FM761765",
        Type : "Motorcycle",
        Year : 2011,
        Make : "Ducati",
        Model : "Monster 1200",
        Color : "Yellow"
```

```
    },
    {
        VIN : "1HVBBAANXWH544237",
        Type : "Semi",
        Year : 2009,
        Make : "Ford",
        Model : "F 150",
        Color : "Black"
    },
    {
        VIN : "1C4RJFAG0FC625797",
        Type : "Sedan",
        Year : 2019,
        Make : "Mercedes",
        Model : "CLK 350",
        Color : "White"
    }
];
export const VEHICLE_REGISTRATION = [
    {
        VIN : "1N4AL11D75C109151",
        LicensePlateNumber : "LEWISR261LL",
        State : "WA",
        City : "Seattle",
        ValidFromDate : new Date("2017-08-21"),
        ValidToDate : new Date("2020-05-11"),
        PendingPenaltyTicketAmount : new Decimal(9025, -2),
        Owners : {
            PrimaryOwner : { PersonId : "" },
            SecondaryOwners : EMPTY_SECONDARY_OWNERS
        }
    },
    {
        VIN : "KM8SRDHF6EU074761",
        LicensePlateNumber : "CA762X",
        State : "WA",
        City : "Kent",
        PendingPenaltyTicketAmount : new Decimal(13075, -2),
        ValidFromDate : new Date("2017-09-14"),
        ValidToDate : new Date("2020-06-25"),
        Owners : {
            PrimaryOwner : { PersonId : "" },
            SecondaryOwners : EMPTY_SECONDARY_OWNERS
        }
```

```
    },
    {
        VIN : "3HGGK5G53FM761765",
        LicensePlateNumber : "CD820Z",
        State : "WA",
        City : "Everett",
        PendingPenaltyTicketAmount : new Decimal(44230, -2),
        ValidFromDate : new Date("2011-03-17"),
        ValidToDate : new Date("2021-03-24"),
        Owners : {
            PrimaryOwner : { PersonId : "" },
            SecondaryOwners : EMPTY_SECONDARY_OWNERS
        }
    },
    {
        VIN : "1HVBBAANXWH544237",
        LicensePlateNumber : "LS477D",
        State : "WA",
        City : "Tacoma",
        PendingPenaltyTicketAmount : new Decimal(4220, -2),
        ValidFromDate : new Date("2011-10-26"),
        ValidToDate : new Date("2023-09-25"),
        Owners : {
            PrimaryOwner : { PersonId : "" },
            SecondaryOwners : EMPTY_SECONDARY_OWNERS
        }
    },
    {
        VIN : "1C4RJFAG0FC625797",
        LicensePlateNumber : "TH393F",
        State : "WA",
        City : "Olympia",
        PendingPenaltyTicketAmount : new Decimal(3045, -2),
        ValidFromDate : new Date("2013-09-02"),
        ValidToDate : new Date("2024-03-19"),
        Owners : {
            PrimaryOwner : { PersonId : "" },
            SecondaryOwners : EMPTY_SECONDARY_OWNERS
        }
    }
];
```

2. `Util.ts`— Un module utilitaire qui importe à partir du `ion-js` package pour fournir des fonctions d'assistance qui convertissent, analysent et impriment les données Amazon Ion.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { Result, TransactionExecutor } from "amazon-qldb-driver-nodejs";
import { GetBlockResponse, GetDigestResponse, ValueHolder } from "aws-sdk/
clients/qldb";
import {
    Decimal,
    decodeUtf8,
    dom,
    IonTypes,
    makePrettyWriter,
    makeReader,
    Reader,
    Timestamp,
    toBase64,
    Writer
} from "ion-js";

import { error } from "./LogUtil";
```

```
/**
 * TODO: Replace this with json.stringify
 * Returns the string representation of a given BlockResponse.
 * @param blockResponse The BlockResponse to convert to string.
 * @returns The string representation of the supplied BlockResponse.
 */
export function blockResponseToString(blockResponse: GetBlockResponse): string {
    let stringBuilder: string = "";
    if (blockResponse.Block.IonText) {
        stringBuilder = stringBuilder + "Block: " + blockResponse.Block.IonText +
 ", ";
    }
    if (blockResponse.Proof.IonText) {
        stringBuilder = stringBuilder + "Proof: " + blockResponse.Proof.IonText;
    }
    stringBuilder = "{" + stringBuilder + "}";
    const writer: Writer = makePrettyWriter();
    const reader: Reader = makeReader(stringBuilder);
    writer.writeValues(reader);
    return decodeUtf8(writer.getBytes());
}

/**
 * TODO: Replace this with json.stringify
 * Returns the string representation of a given GetDigestResponse.
 * @param digestResponse The GetDigestResponse to convert to string.
 * @returns The string representation of the supplied GetDigestResponse.
 */
export function digestResponseToString(digestResponse: GetDigestResponse): string
 {
    let stringBuilder: string = "";
    if (digestResponse.Digest) {
        stringBuilder += "Digest: " + JSON.stringify(toBase64(<Uint8Array>
 digestResponse.Digest)) + ", ";
    }
    if (digestResponse.DigestTipAddress.IonText) {
        stringBuilder += "DigestTipAddress: " +
 digestResponse.DigestTipAddress.IonText;
    }
    stringBuilder = "{" + stringBuilder + "}";
    const writer: Writer = makePrettyWriter();
    const reader: Reader = makeReader(stringBuilder);
```

```
    writer.writeValues(reader);
    return decodeUtf8(writer.getBytes());
}

/**
 * Get the document IDs from the given table.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param tableName The table name to query.
 * @param field A field to query.
 * @param value The key of the given field.
 * @returns Promise which fulfills with the document ID as a string.
 */
export async function getDocumentId(
    txn: TransactionExecutor,
    tableName: string,
    field: string,
    value: string
): Promise<string> {
    const query: string = `SELECT id FROM ${tableName} AS t BY id WHERE t.
${field} = ?`;
    let documentId: string = undefined;
    await txn.execute(query, value).then((result: Result) => {
        const resultList: dom.Value[] = result.getResultList();
        if (resultList.length === 0) {
            throw new Error(`Unable to retrieve document ID using ${value}.`);
        }
        documentId = resultList[0].get("id").stringValue();

    }).catch((err: any) => {
        error(`Error getting documentId: ${err}`);
    });
    return documentId;
}

/**
 * Sleep for the specified amount of time.
 * @param ms The amount of time to sleep in milliseconds.
 * @returns Promise which fulfills with void.
 */
export function sleep(ms: number): Promise<void> {
    return new Promise(resolve => setTimeout(resolve, ms));
}

/**
```

```
 * Find the value of a given path in an Ion value. The path should contain a blob
 value.
 * @param value The Ion value that contains the journal block attributes.
 * @param path The path to a certain attribute.
 * @returns Uint8Array value of the blob, or null if the attribute cannot be
 found in the Ion value
 *                  or is not of type Blob
 */
export function getBlobValue(value: dom.Value, path: string): Uint8Array | null {
    const attribute: dom.Value = value.get(path);
    if (attribute !== null && attribute.getType() === IonTypes.BLOB) {
        return attribute.uInt8ArrayValue();
    }
    return null;
}

/**
 * TODO: Replace this with json.stringify
 * Returns the string representation of a given ValueHolder.
 * @param valueHolder The ValueHolder to convert to string.
 * @returns The string representation of the supplied ValueHolder.
 */
export function valueHolderToString(valueHolder: ValueHolder): string {
    const stringBuilder: string = `{ IonText: ${valueHolder.IonText}}`;
    const writer: Writer = makePrettyWriter();
    const reader: Reader = makeReader(stringBuilder);
    writer.writeValues(reader);
    return decodeUtf8(writer.getBytes());
}

/**
 * Converts a given value to Ion using the provided writer.
 * @param value The value to convert to Ion.
 * @param ionWriter The Writer to pass the value into.
 * @throws Error: If the given value cannot be converted to Ion.
 */
export function writeValueAsIon(value: any, ionWriter: Writer): void {
    switch (typeof value) {
        case "string":
            ionWriter.writeString(value);
            break;
        case "boolean":
            ionWriter.writeBoolean(value);
            break;
```

```
        case "number":
                ionWriter.writeInt(value);
                break;
        case "object":
            if (Array.isArray(value)) {
                // Object is an array.
                ionWriter.stepIn(IonTypes.LIST);

                for (const element of value) {
                    writeValueAsIon(element, ionWriter);
                }

                ionWriter.stepOut();
            } else if (value instanceof Date) {
                // Object is a Date.
                ionWriter.writeTimestamp(Timestamp.parse(value.toISOString()));
            } else if (value instanceof Decimal) {
                // Object is a Decimal.
                ionWriter.writeDecimal(value);
            } else if (value === null) {
                ionWriter.writeNull(IonTypes.NULL);
            } else {
                // Object is a struct.
                ionWriter.stepIn(IonTypes.STRUCT);

                for (const key of Object.keys(value)) {
                    ionWriter.writeFieldName(key);
                    writeValueAsIon(value[key], ionWriter);
                }
                ionWriter.stepOut();
            }
            break;
        default:
            throw new Error(`Cannot convert to Ion for type: ${(typeof
 value)}.`);
    }
}
```

> **Note**
>
> La `getDocumentId` fonction exécute une requête qui renvoie les identifiants de document attribués par le système à partir d'une table. Pour en savoir plus, veuillez consulter la section Utilisation de la clause BY pour interroger l'ID du document.

2. Utilisez le programme suivant (`InsertDocument.ts`) pour insérer les exemples de données dans vos tables.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QldbDriver, Result, TransactionExecutor } from "amazon-qldb-driver-
nodejs";
import { dom } from "ion-js";

import { getQldbDriver } from "./ConnectToLedger";
import { DRIVERS_LICENSE, PERSON, VEHICLE, VEHICLE_REGISTRATION } from "./model/
SampleData";
```

```
import {
    DRIVERS_LICENSE_TABLE_NAME,
    PERSON_TABLE_NAME,
    VEHICLE_REGISTRATION_TABLE_NAME,
    VEHICLE_TABLE_NAME
} from "./qldb/Constants";
import { error, log } from "./qldb/LogUtil";

/**
 * Insert the given list of documents into a table in a single transaction.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param tableName Name of the table to insert documents into.
 * @param documents List of documents to insert.
 * @returns Promise which fulfills with a {@linkcode Result} object.
 */
export async function insertDocument(
    txn: TransactionExecutor,
    tableName: string,
    documents: object[]
): Promise<Result> {
    const statement: string = `INSERT INTO ${tableName} ?`;
    const result: Result = await txn.execute(statement, documents);
    return result;
}

/**
 * Handle the insertion of documents and updating PersonIds all in a single
 transaction.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @returns Promise which fulfills with void.
 */
async function updateAndInsertDocuments(txn: TransactionExecutor): Promise<void> {
    log("Inserting multiple documents into the 'Person' table...");
    const documentIds: Result = await insertDocument(txn, PERSON_TABLE_NAME,
 PERSON);

    const listOfDocumentIds: dom.Value[] = documentIds.getResultList();
    log("Updating PersonIds for 'DriversLicense' and PrimaryOwner for
 'VehicleRegistration'...");
    updatePersonId(listOfDocumentIds);

    log("Inserting multiple documents into the remaining tables...");
    await Promise.all([
        insertDocument(txn, DRIVERS_LICENSE_TABLE_NAME, DRIVERS_LICENSE),
```

```
        insertDocument(txn, VEHICLE_REGISTRATION_TABLE_NAME, VEHICLE_REGISTRATION),
        insertDocument(txn, VEHICLE_TABLE_NAME, VEHICLE)
    ]);
}

/**
 * Update the PersonId value for DriversLicense records and the PrimaryOwner value
 for VehicleRegistration records.
 * @param documentIds List of document IDs.
 */
export function updatePersonId(documentIds: dom.Value[]): void {
    documentIds.forEach((value: dom.Value, i: number) => {
        const documentId: string = value.get("documentId").stringValue();
        DRIVERS_LICENSE[i].PersonId = documentId;
        VEHICLE_REGISTRATION[i].Owners.PrimaryOwner.PersonId = documentId;
    });
}

/**
 * Insert documents into a table in a QLDB ledger.
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
    try {
        const qldbDriver: QldbDriver = getQldbDriver();
        await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
            await updateAndInsertDocuments(txn);
        });
    } catch (e) {
        error(`Unable to insert documents: ${e}`);
    }
}

if (require.main === module) {
    main();
}
```

Note

- Ce programme montre comment appeler la `execute` fonction avec des valeurs paramétrées. Vous pouvez transmettre des paramètres de données en plus de

l'instruction partiQL que vous souhaitez exécuter. Utilisez un point d'interrogation (?) comme espace réservé à une variable dans votre chaîne de déclaration.

- Si une `INSERT` instruction aboutit, elle renvoie le `id` de chaque document inséré.

3. Pour exécuter le programme transpilé, entrez la commande suivante.

```
node dist/InsertDocument.js
```

Vous pouvez ensuite utiliser `SELECT` des instructions pour lire les données des tables du `vehicle-registration` registre. Passez à Étape 4 : interroger les tables d'un registre.

Étape 4 : interroger les tables d'un registre

Après avoir créé des tables dans un registre Amazon QLDB et les avoir chargées avec des données, vous pouvez exécuter des requêtes pour vérifier les données d'immatriculation du véhicule que vous venez d'insérer. QLDB utilise partiQL comme langage de requête et Amazon Ion comme modèle de données orienté document.

partiQL est un langage de requête open source compatible avec SQL qui a été étendu pour fonctionner avec Ion. Avec partiQL, vous pouvez insérer, interroger et gérer vos données à l'aide d'opérateurs SQL courants. Amazon Ion est un sur-ensemble de JSON. Ion est un format de données open source basé sur des documents qui vous donne la flexibilité de stocker et de traiter des données structurées, semi-structurées et imbriquées.

Au cours de cette étape, vous utilisez `SELECT` des instructions pour lire les données des tables du `vehicle-registration` registre.

> **Warning**
>
> Lorsque vous exécutez une requête dans QLDB sans recherche indexée, une analyse complète de la table est déclenchée. partiQL prend en charge ces requêtes car il est compatible avec SQL. Cependant, n'exécutez pas d'analyses de tables pour les cas d'utilisation en production dans QLDB. L'analyse des tables peut entraîner des problèmes de performance sur les tables de grande taille, notamment des conflits de simultanéité et des délais d'expiration des transactions.
>
> Pour éviter de scanner des tables, vous devez exécuter des instructions contenant une clause de `WHERE` prédicat à l'aide d'un opérateur d'égalité sur un champ indexé ou un identifiant de document ; par exemple, `WHERE indexedField = 123` ou. `WHERE`

`indexedField IN (456, 789)` Pour plus d'informations, consultez Optimisation des performances des données.

Pour interroger les tables

1. Utilisez le programme suivant (`FindVehicles.ts`) pour rechercher tous les véhicules enregistrés sous le nom d'une personne dans votre registre.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QldbDriver, Result, TransactionExecutor } from "amazon-qldb-driver-
nodejs";
import { dom } from "ion-js";

import { getQldbDriver } from "./ConnectToLedger";
import { PERSON } from "./model/SampleData";
import { PERSON_TABLE_NAME } from "./qldb/Constants";
import { error, log } from "./qldb/LogUtil";
import { getDocumentId } from "./qldb/Util";
```

```
import { prettyPrintResultList } from "./ScanTable";

/**
 * Query 'Vehicle' and 'VehicleRegistration' tables using a unique document ID in
 one transaction.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param govId The owner's government ID.
 * @returns Promise which fulfills with void.
 */
async function findVehiclesForOwner(txn: TransactionExecutor, govId: string):
 Promise<void> {
    const documentId: string = await getDocumentId(txn, PERSON_TABLE_NAME, "GovId",
 govId);
    const query: string = "SELECT Vehicle FROM Vehicle INNER JOIN
 VehicleRegistration AS r " +
                          "ON Vehicle.VIN = r.VIN WHERE
 r.Owners.PrimaryOwner.PersonId = ?";

    await txn.execute(query, documentId).then((result: Result) => {
        const resultList: dom.Value[] = result.getResultList();
        log(`List of vehicles for owner with GovId: ${govId}`);
        prettyPrintResultList(resultList);
    });
}

/**
 * Find all vehicles registered under a person.
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
    try {
        const qldbDriver: QldbDriver = getQldbDriver();
        await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
            await findVehiclesForOwner(txn, PERSON[0].GovId);
        });
    } catch (e) {
        error(`Error getting vehicles for owner: ${e}`);
    }
}

if (require.main === module) {
    main();
}
```

> **Note**
>
> Tout d'abord, ce programme interroge la `Person` table du document pour `GovId` `LEWISR261LL` obtenir son champ de `id` métadonnées.
> Il utilise ensuite ce document `id` comme clé étrangère pour interroger la `VehicleRegistration` table`PrimaryOwner.PersonId`. Il se joint également `VehicleRegistration` à la `Vehicle` table sur le `VIN` terrain.

2. Pour exécuter le programme transpilé, entrez la commande suivante.

```
node dist/FindVehicles.js
```

Pour en savoir plus sur la modification de documents dans les tables du `vehicle-registration` grand livre, voirÉtape 5 : Modifier les documents d'un registre.

Étape 5 : Modifier les documents d'un registre

Maintenant que vous disposez de données sur lesquelles travailler, vous pouvez commencer à apporter des modifications aux documents du `vehicle-registration` registre dans Amazon QLDB. Dans cette étape, les exemples de code suivants montrent comment exécuter des instructions DML (Data Manipulation Language). Ces déclarations mettent à jour le propriétaire principal d'un véhicule et ajoutent un propriétaire secondaire à un autre véhicule.

Pour modifier des documents

1. Utilisez le programme suivant (`TransferVehicleOwnership.ts`) pour mettre à jour le numéro VIN du propriétaire principal du véhicule `1N4AL11D75C109151` dans votre registre.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
```

```
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QldbDriver, Result, TransactionExecutor } from "amazon-qldb-driver-
nodejs";
import { dom } from "ion-js";

import { getQldbDriver } from "./ConnectToLedger";
import { PERSON, VEHICLE } from "./model/SampleData";
import { PERSON_TABLE_NAME } from "./qldb/Constants";
import { error, log } from "./qldb/LogUtil";
import { getDocumentId } from "./qldb/Util";

/**
 * Query a driver's information using the given ID.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param documentId The unique ID of a document in the Person table.
 * @returns Promise which fulfills with an Ion value containing the person.
 */
export async function findPersonFromDocumentId(txn: TransactionExecutor,
 documentId: string): Promise<dom.Value> {
    const query: string = "SELECT p.* FROM Person AS p BY pid WHERE pid = ?";

    let personId: dom.Value;
    await txn.execute(query, documentId).then((result: Result) => {
        const resultList: dom.Value[] = result.getResultList();
        if (resultList.length === 0) {
            throw new Error(`Unable to find person with ID: ${documentId}.`);
        }
        personId = resultList[0];
    });
    return personId;
```

```
}

/**
 * Find the primary owner for the given VIN.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param vin The VIN to find primary owner for.
 * @returns Promise which fulfills with an Ion value containing the primary owner.
 */
export async function findPrimaryOwnerForVehicle(txn: TransactionExecutor, vin:
 string): Promise<dom.Value> {
    log(`Finding primary owner for vehicle with VIN: ${vin}`);
    const query: string = "SELECT Owners.PrimaryOwner.PersonId FROM
 VehicleRegistration AS v WHERE v.VIN = ?";

    let documentId: string = undefined;
    await txn.execute(query, vin).then((result: Result) => {
        const resultList: dom.Value[] = result.getResultList();
        if (resultList.length === 0) {
            throw new Error(`Unable to retrieve document ID using ${vin}.`);
        }
        const PersonIdValue: dom.Value = resultList[0].get("PersonId");
        if (PersonIdValue === null) {
            throw new Error(`Expected field name PersonId not found.`);
        }
        documentId = PersonIdValue.stringValue();
    });
    return findPersonFromDocumentId(txn, documentId);
}

/**
 * Update the primary owner for a vehicle using the given VIN.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param vin The VIN for the vehicle to operate on.
 * @param documentId New PersonId for the primary owner.
 * @returns Promise which fulfills with void.
 */
async function updateVehicleRegistration(txn: TransactionExecutor, vin: string,
 documentId: string): Promise<void> {
    const statement: string = "UPDATE VehicleRegistration AS r SET
 r.Owners.PrimaryOwner.PersonId = ? WHERE r.VIN = ?";

    log(`Updating the primary owner for vehicle with VIN: ${vin}...`);
    await txn.execute(statement, documentId, vin).then((result: Result) => {
        const resultList: dom.Value[] = result.getResultList();
```

```
        if (resultList.length === 0) {
            throw new Error("Unable to transfer vehicle, could not find
 registration.");
        }
        log(`Successfully transferred vehicle with VIN ${vin} to new owner.`);
    });
}

/**
 * Validate the current owner of the given vehicle and transfer its ownership to a
 new owner in a single transaction.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param vin The VIN of the vehicle to transfer ownership of.
 * @param currentOwner The GovId of the current owner of the vehicle.
 * @param newOwner The GovId of the new owner of the vehicle.
 */
export async function validateAndUpdateRegistration(
    txn: TransactionExecutor,
    vin: string,
    currentOwner: string,
    newOwner: string
): Promise<void> {
    const primaryOwner: dom.Value = await findPrimaryOwnerForVehicle(txn, vin);
    const govIdValue: dom.Value = primaryOwner.get("GovId");
    if (govIdValue !== null && govIdValue.stringValue() !== currentOwner) {
        log("Incorrect primary owner identified for vehicle, unable to transfer.");
    }
    else {
        const documentId: string = await getDocumentId(txn, PERSON_TABLE_NAME,
 "GovId", newOwner);
        await updateVehicleRegistration(txn, vin, documentId);
        log("Successfully transferred vehicle ownership!");
    }
}

/**
 * Find primary owner for a particular vehicle's VIN.
 * Transfer to another primary owner for a particular vehicle's VIN.
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
    try {
        const qldbDriver: QldbDriver = getQldbDriver();
```

```
        const vin: string = VEHICLE[0].VIN;
        const previousOwnerGovId: string = PERSON[0].GovId;
        const newPrimaryOwnerGovId: string = PERSON[1].GovId;

        await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
            await validateAndUpdateRegistration(txn, vin, previousOwnerGovId,
 newPrimaryOwnerGovId);
        });
    } catch (e) {
        error(`Unable to connect and run queries: ${e}`);
    }
}

if (require.main === module) {
    main();
}
```

2. Pour exécuter le programme transpilé, entrez la commande suivante.

```
node dist/TransferVehicleOwnership.js
```

3. Utilisez le programme suivant (`AddSecondaryOwner.ts`) pour ajouter un propriétaire secondaire au véhicule dont le VIN figure `KM8SRDHF6EU074761` dans votre registre.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
```

```
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QldbDriver, Result, TransactionExecutor } from "amazon-qldb-driver-
nodejs";
import { dom } from "ion-js";

import { getQldbDriver } from "./ConnectToLedger";
import { PERSON, VEHICLE_REGISTRATION } from "./model/SampleData";
import { PERSON_TABLE_NAME } from "./qldb/Constants";
import { error, log } from "./qldb/LogUtil";
import { getDocumentId } from "./qldb/Util";
import { prettyPrintResultList } from "./ScanTable";

/**
 * Add a secondary owner into 'VehicleRegistration' table for a particular VIN.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param vin VIN of the vehicle to query.
 * @param secondaryOwnerId The secondary owner's person ID.
 * @returns Promise which fulfills with void.
 */
export async function addSecondaryOwner(
    txn: TransactionExecutor,
    vin: string,
    secondaryOwnerId: string
): Promise<void> {
    log(`Inserting secondary owner for vehicle with VIN: ${vin}`);
    const query: string =
        `FROM VehicleRegistration AS v WHERE v.VIN = ? INSERT INTO
 v.Owners.SecondaryOwners VALUE ?`;

    const personToInsert = {PersonId: secondaryOwnerId};
    await txn.execute(query, vin, personToInsert).then(async (result: Result) => {
        const resultList: dom.Value[] = result.getResultList();
        log("VehicleRegistration Document IDs which had secondary owners added: ");
        prettyPrintResultList(resultList);
    });
}

/**
 * Query for a document ID with a government ID.
```

```
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param governmentId The government ID to query with.
 * @returns Promise which fulfills with the document ID as a string.
 */
export async function getDocumentIdByGovId(txn: TransactionExecutor, governmentId:
 string): Promise<string> {
    const documentId: string = await getDocumentId(txn, PERSON_TABLE_NAME, "GovId",
 governmentId);
    return documentId;
}

/**
 * Check whether a driver has already been registered for the given VIN.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param vin VIN of the vehicle to query.
 * @param secondaryOwnerId The secondary owner's person ID.
 * @returns Promise which fulfills with a boolean.
 */
export async function isSecondaryOwnerForVehicle(
    txn: TransactionExecutor,
    vin: string,
    secondaryOwnerId: string
): Promise<boolean> {
    log(`Finding secondary owners for vehicle with VIN: ${vin}`);
    const query: string = "SELECT Owners.SecondaryOwners FROM VehicleRegistration
 AS v WHERE v.VIN = ?";

    let doesExist: boolean = false;

    await txn.execute(query, vin).then((result: Result) => {
        const resultList: dom.Value[] = result.getResultList();

        resultList.forEach((value: dom.Value) => {
            const secondaryOwnersList: dom.Value[] =
 value.get("SecondaryOwners").elements();

            secondaryOwnersList.forEach((secondaryOwner) => {
                const personId: dom.Value = secondaryOwner.get("PersonId");
                if (personId !== null &&  personId.stringValue() ===
 secondaryOwnerId) {
                    doesExist = true;
                }
            });
        });
```

```
    });
    return doesExist;
}

/**
 * Finds and adds secondary owners for a vehicle.
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
    try {
        const qldbDriver: QldbDriver = getQldbDriver();
        const vin: string = VEHICLE_REGISTRATION[1].VIN;
        const govId: string = PERSON[0].GovId;

        await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
            const documentId: string = await getDocumentIdByGovId(txn, govId);

            if (await isSecondaryOwnerForVehicle(txn, vin, documentId)) {
                log(`Person with ID ${documentId} has already been added as a
 secondary owner of this vehicle.`);
            } else {
                await addSecondaryOwner(txn, vin, documentId);
            }
        });

        log("Secondary owners successfully updated.");
    } catch (e) {
        error(`Unable to add secondary owner: ${e}`);
    }
}

if (require.main === module) {
    main();
}
```

4. Pour exécuter le programme transpilé, entrez la commande suivante.

```
node dist/AddSecondaryOwner.js
```

Pour consulter ces modifications dans le `vehicle-registration` registre, voirÉtape 6 : Afficher l'historique des révisions d'un document.

Étape 6 : Afficher l'historique des révisions d'un document

Après avoir modifié les données d'immatriculation d'un véhicule à l'étape précédente, vous pouvez consulter l'historique de tous ses propriétaires enregistrés et tout autre champ mis à jour. Au cours de cette étape, vous recherchez l'historique des révisions d'un document dans le `VehicleRegistration` tableau de votre `vehicle-registration` grand livre.

Pour consulter l'historique des révisions

1. Utilisez le programme suivant (`QueryHistory.ts`) pour rechercher l'historique des révisions du `VehicleRegistration` document avec le VIN`1N4AL11D75C109151`.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QldbDriver, Result, TransactionExecutor } from "amazon-qldb-driver-
nodejs";
import { dom } from "ion-js";

import { getQldbDriver } from "./ConnectToLedger";
import { VEHICLE_REGISTRATION } from "./model/SampleData";
```

```
import { VEHICLE_REGISTRATION_TABLE_NAME } from "./qldb/Constants";
import { prettyPrintResultList } from "./ScanTable";
import { error, log } from "./qldb/LogUtil";
import { getDocumentId } from "./qldb/Util";

/**
 * Find previous primary owners for the given VIN in a single transaction.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param vin The VIN to find previous primary owners for.
 * @returns Promise which fulfills with void.
 */
async function previousPrimaryOwners(txn: TransactionExecutor, vin: string):
 Promise<void> {
    const documentId: string = await getDocumentId(txn,
 VEHICLE_REGISTRATION_TABLE_NAME, "VIN", vin);
    const todaysDate: Date = new Date();
    // set todaysDate back one minute to ensure end time is in the past
    // by the time the request reaches our backend
    todaysDate.setMinutes(todaysDate.getMinutes() - 1);
    const threeMonthsAgo: Date = new Date(todaysDate);
    threeMonthsAgo.setMonth(todaysDate.getMonth() - 3);

    const query: string =
        `SELECT data.Owners.PrimaryOwner, metadata.version FROM history ` +
        `(${VEHICLE_REGISTRATION_TABLE_NAME}, \`${threeMonthsAgo.toISOString()}\`,
 \`${todaysDate.toISOString()}\`) ` +
        `AS h WHERE h.metadata.id = ?`;

    await txn.execute(query, documentId).then((result: Result) => {
        log(`Querying the 'VehicleRegistration' table's history using VIN:
 ${vin}.`);
        const resultList: dom.Value[] = result.getResultList();
        prettyPrintResultList(resultList);
    });
}

/**
 * Query a table's history for a particular set of documents.
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
    try {
        const qldbDriver: QldbDriver = getQldbDriver();
        const vin: string = VEHICLE_REGISTRATION[0].VIN;
```

```
        await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
            await previousPrimaryOwners(txn, vin);
        });
    } catch (e) {
        error(`Unable to query history to find previous owners: ${e}`);
    }
}

if (require.main === module) {
    main();
}
```

> **Note**
>
> - Vous pouvez consulter l'historique des révisions d'un document en interrogeant la syntaxe intégrée Fonction historique ci-dessous.
>
>   ```
>   SELECT * FROM history( table_name [, `start-time` [, `end-time` ] ] ) AS h
>   [ WHERE h.metadata.id = 'id' ]
>   ```
>
> - L'heure de début et l'heure de fin sont toutes deux facultatives. Ce sont des valeurs littérales d'Amazon Ion qui peuvent être indiquées par des backticks (). `` `...` `` Pour plus d'informations, consultez Interroger Ion avec PartiQL dans Amazon QLDB.
> - Il est recommandé de qualifier une requête d'historique à la fois par une plage de dates (heure de début et heure de fin) et par un identifiant de document (`metadata.id`). QLDB `SELECT` traite les requêtes dans les transactions, qui sont soumises à une limite de délai d'expiration des transactions.
>
>   L'historique QLDB est indexé par ID de document, et vous ne pouvez pas créer d'index d'historique supplémentaire pour le moment. Les requêtes d'historique qui incluent une heure de début et une heure de fin bénéficient de la qualification par plage de dates.

2. Pour exécuter le programme transpilé, entrez la commande suivante.

```
node dist/QueryHistory.js
```

Pour vérifier cryptographiquement une révision de document dans le `vehicle-registration` registre, passez à. Étape 7 : Vérifier un document dans un registre

Étape 7 : Vérifier un document dans un registre

Avec Amazon QLDB, vous pouvez vérifier efficacement l'intégrité d'un document dans le journal de votre grand livre en utilisant le hachage cryptographique avec SHA-256. Pour en savoir plus sur le fonctionnement de la vérification et du hachage cryptographique dans QLDB, consultez. Vérification des données dans Amazon QLDB

Au cours de cette étape, vous devez vérifier une révision de document dans le `VehicleRegistration` tableau de votre `vehicle-registration` grand livre. Tout d'abord, vous demandez un résumé, qui est renvoyé sous forme de fichier de sortie et sert de signature à l'historique complet des modifications de votre registre. Ensuite, vous demandez une preuve pour la révision relative à ce condensé. À l'aide de cette preuve, l'intégrité de votre révision est vérifiée si tous les contrôles de validation sont réussis.

Pour vérifier la révision d'un document

1. Passez en revue les `.ts` fichiers suivants, qui contiennent les objets QLDB requis pour la vérification.

   1. `BlockAddress.ts`

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
```

```
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { ValueHolder } from "aws-sdk/clients/qldb";
import { dom, IonTypes } from "ion-js";

export class BlockAddress {
    _strandId: string;
    _sequenceNo: number;

    constructor(strandId: string, sequenceNo: number) {
        this._strandId = strandId;
        this._sequenceNo = sequenceNo;
    }
}

/**
 * Convert a block address from an Ion value into a ValueHolder.
 * Shape of the ValueHolder must be: {'IonText': "{strandId: <"strandId">,
 sequenceNo: <sequenceNo>}"}
 * @param value The Ion value that contains the block address values to convert.
 * @returns The ValueHolder that contains the strandId and sequenceNo.
 */
export function blockAddressToValueHolder(value: dom.Value): ValueHolder {
    const blockAddressValue : dom.Value = getBlockAddressValue(value);
    const strandId: string = getStrandId(blockAddressValue);
    const sequenceNo: number = getSequenceNo(blockAddressValue);
    const valueHolder: string = `{strandId: "${strandId}", sequenceNo:
 ${sequenceNo}}`;
    const blockAddress: ValueHolder = {IonText: valueHolder};
    return blockAddress;
}

/**
 * Helper method that to get the Metadata ID.
 * @param value The Ion value.
 * @returns The Metadata ID.
 */
export function getMetadataId(value: dom.Value): string {
    const metaDataId: dom.Value = value.get("id");
    if (metaDataId === null) {
        throw new Error(`Expected field name id, but not found.`);
    }
```

```
    return metaDataId.stringValue();
}

/**
 * Helper method to get the Sequence No.
 * @param value The Ion value.
 * @returns The Sequence No.
 */
export function getSequenceNo(value : dom.Value): number {
    const sequenceNo: dom.Value = value.get("sequenceNo");
    if (sequenceNo === null) {
        throw new Error(`Expected field name sequenceNo, but not found.`);
    }
    return sequenceNo.numberValue();
}

/**
 * Helper method to get the Strand ID.
 * @param value The Ion value.
 * @returns The Strand ID.
 */
export function getStrandId(value: dom.Value): string {
    const strandId: dom.Value = value.get("strandId");
    if (strandId === null) {
        throw new Error(`Expected field name strandId, but not found.`);
    }
    return strandId.stringValue();
}

export function getBlockAddressValue(value: dom.Value) : dom.Value {
    const type = value.getType();
    if (type !== IonTypes.STRUCT) {
        throw new Error(`Unexpected format: expected struct, but got IonType:
 ${type.name}`);
    }
    const blockAddress: dom.Value = value.get("blockAddress");
    if (blockAddress == null) {
        throw new Error(`Expected field name blockAddress, but not found.`);
    }
    return blockAddress;
}
```

2. Verifier.ts

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { Digest, ValueHolder } from "aws-sdk/clients/qldb";
import { createHash } from "crypto";
import { dom, toBase64 } from "ion-js";

import { getBlobValue } from "./Util";

const HASH_LENGTH: number = 32;
const UPPER_BOUND: number = 8;

/**
 * Build the candidate digest representing the entire ledger from the Proof
 hashes.
 * @param proof The Proof object.
 * @param leafHash The revision hash to pair with the first hash in the Proof
 hashes list.
 * @returns The calculated root hash.
 */
```

```
function buildCandidateDigest(proof: ValueHolder, leafHash: Uint8Array):
 Uint8Array {
    const parsedProof: Uint8Array[] = parseProof(proof);
    const rootHash: Uint8Array = calculateRootHashFromInternalHash(parsedProof,
 leafHash);
    return rootHash;
}

/**
 * Combine the internal hashes and the leaf hash until only one root hash
 remains.
 * @param internalHashes An array of hash values.
 * @param leafHash The revision hash to pair with the first hash in the Proof
 hashes list.
 * @returns The root hash constructed by combining internal hashes.
 */
function calculateRootHashFromInternalHash(internalHashes: Uint8Array[],
 leafHash: Uint8Array): Uint8Array {
    const rootHash: Uint8Array = internalHashes.reduce(joinHashesPairwise,
 leafHash);
    return rootHash;
}

/**
 * Compare two hash values by converting each Uint8Array byte, which is unsigned
 by default,
 * into a signed byte, assuming they are little endian.
 * @param hash1 The hash value to compare.
 * @param hash2 The hash value to compare.
 * @returns Zero if the hash values are equal, otherwise return the difference of
 the first pair of non-matching bytes.
 */
function compareHashValues(hash1: Uint8Array, hash2: Uint8Array): number {
    if (hash1.length !== HASH_LENGTH || hash2.length !== HASH_LENGTH) {
        throw new Error("Invalid hash.");
    }
    for (let i = hash1.length-1; i >= 0; i--) {
        const difference: number = (hash1[i]<<24 >>24) - (hash2[i]<<24 >>24);
        if (difference !== 0) {
            return difference;
        }
    }
    return 0;
}
```

```
/**
 * Helper method that concatenates two Uint8Array.
 * @param arrays List of array to concatenate, in the order provided.
 * @returns The concatenated array.
 */
function concatenate(...arrays: Uint8Array[]): Uint8Array {
    let totalLength = 0;
    for (const arr of arrays) {
        totalLength += arr.length;
    }
    const result = new Uint8Array(totalLength);
    let offset = 0;
    for (const arr of arrays) {
        result.set(arr, offset);
        offset += arr.length;
    }
    return result;
}

/**
 * Flip a single random bit in the given hash value.
 * This method is intended to be used for purpose of demonstrating the QLDB
 verification features only.
 * @param original The hash value to alter.
 * @returns The altered hash with a single random bit changed.
 */
export function flipRandomBit(original: any): Uint8Array {
    if (original.length === 0) {
        throw new Error("Array cannot be empty!");
    }
    const bytePos: number = Math.floor(Math.random() * original.length);
    const bitShift: number = Math.floor(Math.random() * UPPER_BOUND);
    const alteredHash: Uint8Array = original;

    alteredHash[bytePos] = alteredHash[bytePos] ^ (1 << bitShift);
    return alteredHash;
}

/**
 * Take two hash values, sort them, concatenate them, and generate a new hash
 value from the concatenated values.
 * @param h1 Byte array containing one of the hashes to compare.
 * @param h2 Byte array containing one of the hashes to compare.
```

```
 * @returns The concatenated array of hashes.
 */
export function joinHashesPairwise(h1: Uint8Array, h2: Uint8Array): Uint8Array {
    if (h1.length === 0) {
        return h2;
    }
    if (h2.length === 0) {
        return h1;
    }
    let concat: Uint8Array;
    if (compareHashValues(h1, h2) < 0) {
        concat = concatenate(h1, h2);
    } else {
        concat = concatenate(h2, h1);
    }
    const hash = createHash('sha256');
    hash.update(concat);
    const newDigest: Uint8Array = hash.digest();
    return newDigest;
}

/**
 * Parse the Block object returned by QLDB and retrieve block hash.
 * @param valueHolder A structure containing an Ion string value.
 * @returns The block hash.
 */
export function parseBlock(valueHolder: ValueHolder): Uint8Array {
    const block: dom.Value = dom.load(valueHolder.IonText);
    const blockHash: Uint8Array = getBlobValue(block, "blockHash");
    return blockHash;
}

/**
 * Parse the Proof object returned by QLDB into an iterator.
 * The Proof object returned by QLDB is a dictionary like the following:
 * {'IonText': '[{{<hash>}},{{<hash>}}]'}
 * @param valueHolder A structure containing an Ion string value.
 * @returns A list of hash values.
 */
function parseProof(valueHolder: ValueHolder): Uint8Array[] {
    const proofs : dom.Value = dom.load(valueHolder.IonText);
    return proofs.elements().map(proof => proof.uInt8ArrayValue());
}
```

```
/**
 *  Verify document revision against the provided digest.
 * @param documentHash The SHA-256 value representing the document revision to be
 verified.
 * @param digest The SHA-256 hash value representing the ledger digest.
 * @param proof The Proof object retrieved from GetRevision.getRevision.
 * @returns If the document revision verifies against the ledger digest.
 */
export function verifyDocument(documentHash: Uint8Array, digest: Digest, proof:
 ValueHolder): boolean {
    const candidateDigest = buildCandidateDigest(proof, documentHash);
    return (toBase64(<Uint8Array> digest) === toBase64(candidateDigest));
}
```

2. Utilisez deux `.ts` programmes (`GetDigest.tsetGetRevision.ts`) pour effectuer les étapes suivantes :

   - Demandez un nouveau résumé à partir du `vehicle-registration` registre.
   - Demandez une preuve pour chaque révision du document avec le VIN indiqué `1N4AL11D75C109151` dans le `VehicleRegistration` tableau.
   - Vérifiez les révisions à l'aide du résumé renvoyé et de la preuve en recalculant le résumé.

   Le `GetDigest.ts` programme contient le code suivant.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
```

```
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QLDB } from "aws-sdk";
import { GetDigestRequest, GetDigestResponse } from "aws-sdk/clients/qldb";

import { LEDGER_NAME } from "./qldb/Constants";
import { error, log } from "./qldb/LogUtil";
import { digestResponseToString } from "./qldb/Util";

/**
 * Get the digest of a ledger's journal.
 * @param ledgerName Name of the ledger to operate on.
 * @param qldbClient The QLDB control plane client to use.
 * @returns Promise which fulfills with a GetDigestResponse.
 */
export async function getDigestResult(ledgerName: string, qldbClient: QLDB):
 Promise<GetDigestResponse> {
    const request: GetDigestRequest = {
        Name: ledgerName
    };
    const result: GetDigestResponse = await
 qldbClient.getDigest(request).promise();
    return result;
}

/**
 * This is an example for retrieving the digest of a particular ledger.
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
    try {
        const qldbClient: QLDB = new QLDB();
        log(`Retrieving the current digest for ledger: ${LEDGER_NAME}.`);
        const digest: GetDigestResponse = await getDigestResult(LEDGER_NAME,
 qldbClient);
        log(`Success. Ledger digest: \n${digestResponseToString(digest)}.`);
    } catch (e) {
        error(`Unable to get a ledger digest: ${e}`);
```

```
    }
}

if (require.main === module) {
    main();
}
```

> **Note**
>
> Utilisez cette `getDigest` fonction pour demander un résumé reprenant l'extrait actuel du journal dans votre registre. Le conseil du journal fait référence au dernier bloc validé au moment où QLDB reçoit votre demande.

Le `GetRevision.ts` programme contient le code suivant.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QldbDriver, TransactionExecutor } from "amazon-qldb-driver-nodejs";
```

```
import { QLDB } from "aws-sdk";
import { Digest, GetDigestResponse, GetRevisionRequest, GetRevisionResponse,
 ValueHolder } from "aws-sdk/clients/qldb";
import { dom, toBase64 } from "ion-js";

import { getQldbDriver } from "./ConnectToLedger";
import { getDigestResult } from './GetDigest';
import { VEHICLE_REGISTRATION } from "./model/SampleData"
import { blockAddressToValueHolder, getMetadataId } from './qldb/BlockAddress';
import { LEDGER_NAME } from './qldb/Constants';
import { error, log } from "./qldb/LogUtil";
import { getBlobValue, valueHolderToString } from "./qldb/Util";
import { flipRandomBit, verifyDocument } from "./qldb/Verifier";

/**
 * Get the revision data object for a specified document ID and block address.
 * Also returns a proof of the specified revision for verification.
 * @param ledgerName Name of the ledger containing the document to query.
 * @param documentId Unique ID for the document to be verified, contained in the
 committed view of the document.
 * @param blockAddress The location of the block to request.
 * @param digestTipAddress The latest block location covered by the digest.
 * @param qldbClient The QLDB control plane client to use.
 * @returns Promise which fulfills with a GetRevisionResponse.
 */
async function getRevision(
    ledgerName: string,
    documentId: string,
    blockAddress: ValueHolder,
    digestTipAddress: ValueHolder,
    qldbClient: QLDB
): Promise<GetRevisionResponse> {
    const request: GetRevisionRequest = {
        Name: ledgerName,
        BlockAddress: blockAddress,
        DocumentId: documentId,
        DigestTipAddress: digestTipAddress
    };
    const result: GetRevisionResponse = await
 qldbClient.getRevision(request).promise();
    return result;
}

/**
```

```
 * Query the table metadata for a particular vehicle for verification.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param vin VIN to query the table metadata of a specific registration with.
 * @returns Promise which fulfills with a list of Ion values that contains the
 results of the query.
 */
export async function lookupRegistrationForVin(txn: TransactionExecutor, vin:
 string): Promise<dom.Value[]> {
    log(`Querying the 'VehicleRegistration' table for VIN: ${vin}...`);
    let resultList: dom.Value[];
    const query: string = "SELECT blockAddress, metadata.id FROM
 _ql_committed_VehicleRegistration WHERE data.VIN = ?";

    await txn.execute(query, vin).then(function(result) {
      resultList = result.getResultList();
    });
    return resultList;
}

/**
 * Verify each version of the registration for the given VIN.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param ledgerName The ledger to get the digest from.
 * @param vin VIN to query the revision history of a specific registration with.
 * @param qldbClient The QLDB control plane client to use.
 * @returns Promise which fulfills with void.
 * @throws Error: When verification fails.
 */
export async function verifyRegistration(
    txn: TransactionExecutor,
    ledgerName: string,
    vin: string,
    qldbClient: QLDB
): Promise<void> {
    log(`Let's verify the registration with VIN = ${vin}, in ledger =
 ${ledgerName}.`);
    const digest: GetDigestResponse = await getDigestResult(ledgerName,
 qldbClient);
    const digestBytes: Digest = digest.Digest;
    const digestTipAddress: ValueHolder = digest.DigestTipAddress;

    log(
        `Got a ledger digest: digest tip address = \n
${valueHolderToString(digestTipAddress)},
```

```
        digest = \n${toBase64(<Uint8Array> digestBytes)}.`
    );
    log(`Querying the registration with VIN = ${vin} to verify each version of the
registration...`);
    const resultList: dom.Value[] = await lookupRegistrationForVin(txn, vin);
    log("Getting a proof for the document.");

    for (const result of resultList) {
        const blockAddress: ValueHolder =  blockAddressToValueHolder(result);
        const documentId: string = getMetadataId(result);

        const revisionResponse: GetRevisionResponse = await getRevision(
            ledgerName,
            documentId,
            blockAddress,
            digestTipAddress,
            qldbClient
        );

        const revision: dom.Value = dom.load(revisionResponse.Revision.IonText);
        const documentHash: Uint8Array = getBlobValue(revision, "hash");
        const proof: ValueHolder = revisionResponse.Proof;
        log(`Got back a proof: ${valueHolderToString(proof)}.`);

        let verified: boolean = verifyDocument(documentHash, digestBytes, proof);
        if (!verified) {
           throw new Error("Document revision is not verified.");
        } else {
            log("Success! The document is verified.");
        }
        const alteredDocumentHash: Uint8Array = flipRandomBit(documentHash);

        log(
            `Flipping one bit in the document's hash and assert that the document
is NOT verified.
            The altered document hash is: ${toBase64(alteredDocumentHash)}`
        );
        verified = verifyDocument(alteredDocumentHash, digestBytes, proof);

        if (verified) {
            throw new Error("Expected altered document hash to not be verified
against digest.");
        } else {
```

```
            log("Success! As expected flipping a bit in the document hash causes
 verification to fail.");
        }
        log(`Finished verifying the registration with VIN = ${vin} in ledger =
 ${ledgerName}.`);
    }
}

/**
 * Verify the integrity of a document revision in a QLDB ledger.
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
    try {
        const qldbClient: QLDB = new QLDB();
        const qldbDriver: QldbDriver = getQldbDriver();

        const registration = VEHICLE_REGISTRATION[0];
        const vin: string = registration.VIN;

        await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
            await verifyRegistration(txn, LEDGER_NAME, vin, qldbClient);
        });
    } catch (e) {
        error(`Unable to verify revision: ${e}`);
    }
}

if (require.main === module) {
    main();
}
```

> **Note**
>
> Une fois que la `getRevision` fonction a renvoyé une preuve pour la révision du document spécifiée, ce programme utilise une API côté client pour vérifier cette révision.

3. Pour exécuter le programme transpilé, entrez la commande suivante.

```
node dist/GetRevision.js
```

Si vous n'avez plus besoin d'utiliser le `vehicle-registration` registre, passez àÉtape 8 (optionnelle) : Nettoyer les ressources.

Étape 8 (optionnelle) : Nettoyer les ressources

Vous pouvez continuer à utiliser le `vehicle-registration` registre. Toutefois, si vous n'en avez plus besoin, vous devez le supprimer.

Pour supprimer le registre

1. Utilisez le programme suivant (`DeleteLedger.ts`) pour supprimer votre `vehicle-registration` registre et tout son contenu.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { isResourceNotFoundException } from "amazon-qldb-driver-nodejs";
import { AWSError, QLDB } from "aws-sdk";
import { DeleteLedgerRequest, DescribeLedgerRequest } from "aws-sdk/clients/qldb";

import { setDeletionProtection } from "./DeletionProtection";
```

```
import { LEDGER_NAME } from "./qldb/Constants";
import { error, log } from "./qldb/LogUtil";
import { sleep } from "./qldb/Util";

const LEDGER_DELETION_POLL_PERIOD_MS = 20000;

/**
 * Send a request to QLDB to delete the specified ledger.
 * @param ledgerName Name of the ledger to be deleted.
 * @param qldbClient The QLDB control plane client to use.
 * @returns Promise which fulfills with void.
 */
export async function deleteLedger(ledgerName: string, qldbClient: QLDB):
 Promise<void> {
    log(`Attempting to delete the ledger with name: ${ledgerName}`);
    const request: DeleteLedgerRequest = {
        Name: ledgerName
    };
    await qldbClient.deleteLedger(request).promise();
    log("Success.");
}

/**
 * Wait for the ledger to be deleted.
 * @param ledgerName Name of the ledger to be deleted.
 * @param qldbClient The QLDB control plane client to use.
 * @returns Promise which fulfills with void.
 */
export async function waitForDeleted(ledgerName: string, qldbClient: QLDB):
 Promise<void> {
    log("Waiting for the ledger to be deleted...");
    const request: DescribeLedgerRequest = {
        Name: ledgerName
    };
    let isDeleted: boolean = false;
    while (true) {
        await qldbClient.describeLedger(request).promise().catch((error: AWSError)
 => {
            if (isResourceNotFoundException(error)) {
                isDeleted = true;
                log("Success. Ledger is deleted.");
            }
        });
        if (isDeleted) {
```

```
            break;
        }
        log("The ledger is still being deleted. Please wait...");
        await sleep(LEDGER_DELETION_POLL_PERIOD_MS);
    }
}

/**
 * Delete a ledger.
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
    try {
        const qldbClient: QLDB = new QLDB();
        await setDeletionProtection(LEDGER_NAME, qldbClient, false);
        await deleteLedger(LEDGER_NAME, qldbClient);
        await waitForDeleted(LEDGER_NAME, qldbClient);
    } catch (e) {
        error(`Unable to delete the ledger: ${e}`);
    }
}

if (require.main === module) {
    main();
}
```

> **Note**
>
> Si la protection contre la suppression est activée pour votre registre, vous devez d'abord la désactiver avant de pouvoir le supprimer à l'aide de l'API QLDB.

2. Pour exécuter le programme transpilé, entrez la commande suivante.

```
node dist/DeleteLedger.js
```

Tutoriel Amazon QLDB Python

Dans cette implémentation de l'exemple d'application du didacticiel, vous utilisez le pilote Amazon QLDB avec leAWS SDK for Python (Boto3) pour créer un registre QLDB et le remplir avec des exemples de données.

Lors de l'utilisation de l'APIAWS SDK for Python (Boto3), vous pouvez vous référer au client QLDB (Boto3). Vous pouvez vous référer au client de bas niveau QLDB (Boto3). Pour les opérations sur les données transactionnelles, vous pouvez vous référer au guide de référence de l'API QLDB pour Python.

> **Note**
>
> Le cas échéant, certaines étapes du didacticiel comportent des commandes ou des exemples de code différents pour chaque version majeure prise en charge du pilote QLDB pour Python.

Rubriques

Installation de l'exemple d'application Python Amazon QLDB

Cette section explique comment installer et exécuter l'exemple d'application Amazon QLDB fourni pour le didacticiel step-by-step Python. Le cas d'utilisation de cet exemple d'application est une base de données du ministère des véhicules automobiles (DMV) qui permet de suivre l'historique complet des immatriculations de véhicules.

L'exemple d'application DMV pour Python est open source dans le GitHub référentiel aws-samples/amazon-qldb-dmv-sample -python.

Prérequis

Avant de commencer, assurez-vous d'avoir terminé le pilote QLDB for PythonPrérequis. Cela inclut l'installation de Python et les opérations suivantes :

1. Inscrivez-vous àAWS.
2. Créez un utilisateur doté des autorisations QLDB appropriées.
3. Accorder un accès programmatique pour le développement.

Pour effectuer toutes les étapes de ce didacticiel, vous devez disposer d'un accès administratif complet à votre ressource de registre via l'API QLDB.

Installation

Pour installer l'exemple d'application

1. Entrez la`pip` commande suivante pour cloner et installer l'exemple d'application à partir de celui-ci GitHub.

   3.x

   ```
   pip install git+https://github.com/aws-samples/amazon-qldb-dmv-sample-python.git
   ```

   2.x

   ```
   pip install git+https://github.com/aws-samples/amazon-qldb-dmv-sample-
   python.git@v1.0.0
   ```

   L'exemple d'application regroupe le code source complet de ce didacticiel et ses dépendances, y compris le pilote Python et le AWS SDK for Python (Boto3).
2. Avant de commencer à exécuter du code sur la ligne de commande, basculez votre répertoire de travail actuel vers l'emplacement où le`pyqldbsamples` package est installé. Entrez la commande suivante.

   ```
   cd $(python -c "import pyqldbsamples; print(pyqldbsamples.__path__[0])")
   ```
3. PassezÉtape 1 : Créer un nouveau registre à démarrer le didacticiel et à créer un registre.

Étape 1 : Créer un nouveau registre

Au cours de cette étape, vous allez créer un nouveau registre Amazon QLDB nommé`vehicle-registration`.

Pour créer un nouveau registre

1. Consultez le fichier suivant (`constants.py`), qui contient des valeurs constantes utilisées par tous les autres programmes de ce didacticiel.

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.


class Constants:
    """
    Constant values used throughout this tutorial.
    """
    LEDGER_NAME = "vehicle-registration"

    VEHICLE_REGISTRATION_TABLE_NAME = "VehicleRegistration"
    VEHICLE_TABLE_NAME = "Vehicle"
    PERSON_TABLE_NAME = "Person"
    DRIVERS_LICENSE_TABLE_NAME = "DriversLicense"

    LICENSE_NUMBER_INDEX_NAME = "LicenseNumber"
    GOV_ID_INDEX_NAME = "GovId"
    VEHICLE_VIN_INDEX_NAME = "VIN"
    LICENSE_PLATE_NUMBER_INDEX_NAME = "LicensePlateNumber"
```

```
    PERSON_ID_INDEX_NAME = "PersonId"

    JOURNAL_EXPORT_S3_BUCKET_NAME_PREFIX = "qldb-tutorial-journal-export"
    USER_TABLES = "information_schema.user_tables"
    S3_BUCKET_ARN_TEMPLATE = "arn:aws:s3:::"
    LEDGER_NAME_WITH_TAGS = "tags"

    RETRY_LIMIT = 4
```

2. Utilisez le programme suivant (`create_ledger.py`) pour créer un registre nommé`vehicle-registration`.

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO
from time import sleep

from boto3 import client

from pyqldbsamples.constants import Constants
```

```
logger = getLogger(__name__)
basicConfig(level=INFO)
qldb_client = client('qldb')

LEDGER_CREATION_POLL_PERIOD_SEC = 10
ACTIVE_STATE = "ACTIVE"


def create_ledger(name):
    """
    Create a new ledger with the specified name.

    :type name: str
    :param name: Name for the ledger to be created.

    :rtype: dict
    :return: Result from the request.
    """
    logger.info("Let's create the ledger named: {}...".format(name))
    result = qldb_client.create_ledger(Name=name, PermissionsMode='ALLOW_ALL')
    logger.info('Success. Ledger state: {}.'.format(result.get('State')))
    return result


def wait_for_active(name):
    """
    Wait for the newly created ledger to become active.

    :type name: str
    :param name: The ledger to check on.

    :rtype: dict
    :return: Result from the request.
    """
    logger.info('Waiting for ledger to become active...')
    while True:
        result = qldb_client.describe_ledger(Name=name)
        if result.get('State') == ACTIVE_STATE:
            logger.info('Success. Ledger is active and ready to use.')
            return result
        logger.info('The ledger is still creating. Please wait...')
        sleep(LEDGER_CREATION_POLL_PERIOD_SEC)
```

```
def main(ledger_name=Constants.LEDGER_NAME):
    """
    Create a ledger and wait for it to be active.
    """
    try:
        create_ledger(ledger_name)
        wait_for_active(ledger_name)
    except Exception as e:
        logger.exception('Unable to create the ledger!')
        raise e


if __name__ == '__main__':
    main()
```

> **Note**
>
> - Lors de l'`create_ledger`appel, vous devez spécifier un nom de registre et un mode d'autorisation. Nous vous recommandons d'utiliser le mode d'`STANDARD`autorisations pour optimiser la sécurité des données de votre registre.
> - Lors de la création d'un registre, la protection contre la suppression est activée par défaut. Il s'agit d'une fonctionnalité de QLDB qui empêche la suppression de registres par n'importe quel utilisateur. Vous avez la possibilité de désactiver la protection contre la suppression lors de la création d'un registre à l'aide de l'API QLDB ou duAWS Command Line Interface (AWS CLI).
> - En option, vous pouvez également spécifier des étiquettes à associer à votre registre.

3. Pour exécuter le programme, saisissez la commande suivante.

```
python create_ledger.py
```

Pour vérifier votre connexion au nouveau registre, passez àÉtape 2 : testez la connectivité au registre.

Étape 2 : testez la connectivité au registre

Au cours de cette étape, vous devez vérifier que vous pouvez vous connecter au`vehicle-registration` registre dans Amazon QLDB à l'aide du point de terminaison de l'API de données transactionnelles.

Pour tester la connectivité au registre

1. Utilisez le programme suivant (`connect_to_ledger.py`) pour créer une connexion de session de données au`vehicle-registration` registre.

   3.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from botocore.exceptions import ClientError

from pyqldb.driver.qldb_driver import QldbDriver
```

```
from pyqldbsamples.constants import Constants

logger = getLogger(__name__)
basicConfig(level=INFO)


def create_qldb_driver(ledger_name=Constants.LEDGER_NAME, region_name=None,
 endpoint_url=None, boto3_session=None):
    """
    Create a QLDB driver for executing transactions.

    :type ledger_name: str
    :param ledger_name: The QLDB ledger name.

    :type region_name: str
    :param region_name: See [1].

    :type endpoint_url: str
    :param endpoint_url: See [1].

    :type boto3_session: :py:class:`boto3.session.Session`
    :param boto3_session: The boto3 session to create the client with (see [1]).

    :rtype: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :return: A QLDB driver object.

    [1]: `Boto3 Session.client Reference <https://
boto3.amazonaws.com/v1/documentation/api/latest/reference/core/
session.html#boto3.session.Session.client>`.
    """
    qldb_driver = QldbDriver(ledger_name=ledger_name, region_name=region_name,
 endpoint_url=endpoint_url,
                             boto3_session=boto3_session)
    return qldb_driver


def main(ledger_name=Constants.LEDGER_NAME):
    """
    Connect to a given ledger using default settings.
    """
    try:
        with create_qldb_driver(ledger_name) as driver:
            logger.info('Listing table names ')
            for table in driver.list_tables():
```

```
                logger.info(table)
    except ClientError as ce:
        logger.exception('Unable to list tables.')
        raise ce


if __name__ == '__main__':
    main()
```

> **Note**
>
> - Pour exécuter des transactions de données sur votre registre, vous devez créer un objet pilote QLDB pour vous connecter à un registre spécifique. Il s'agit d'un objet client différent de `qldb_client` celui que vous avez utilisé à l'étape précédente pour créer le registre. Ce client précédent est uniquement utilisé pour exécuter les opérations de l'API de gestion répertoriées dans le Référence d'API Amazon QLDB.
> - Vous devez spécifier un nom de registre lorsque vous créez cet objet pilote.

2.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
```

```
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from botocore.exceptions import ClientError

from pyqldb.driver.pooled_qldb_driver import PooledQldbDriver
from pyqldbsamples.constants import Constants

logger = getLogger(__name__)
basicConfig(level=INFO)


def create_qldb_driver(ledger_name=Constants.LEDGER_NAME, region_name=None,
 endpoint_url=None, boto3_session=None):
    """
    Create a QLDB driver for creating sessions.

    :type ledger_name: str
    :param ledger_name: The QLDB ledger name.

    :type region_name: str
    :param region_name: See [1].

    :type endpoint_url: str
    :param endpoint_url: See [1].

    :type boto3_session: :py:class:`boto3.session.Session`
    :param boto3_session: The boto3 session to create the client with (see [1]).

    :rtype: :py:class:`pyqldb.driver.pooled_qldb_driver.PooledQldbDriver`
    :return: A pooled QLDB driver object.

    [1]: `Boto3 Session.client Reference <https://
boto3.amazonaws.com/v1/documentation/api/latest/reference/core/
session.html#boto3.session.Session.client>`.
    """
    qldb_driver = PooledQldbDriver(ledger_name=ledger_name,
 region_name=region_name, endpoint_url=endpoint_url,
```

```
                                    boto3_session=boto3_session)
    return qldb_driver


def create_qldb_session():
    """
    Retrieve a QLDB session object.

    :rtype: :py:class:`pyqldb.session.pooled_qldb_session.PooledQldbSession`
    :return: A pooled QLDB session object.
    """
    qldb_session = pooled_qldb_driver.get_session()
    return qldb_session


pooled_qldb_driver = create_qldb_driver()


if __name__ == '__main__':
    """
    Connect to a session for a given ledger using default settings.
    """
    try:
        qldb_session = create_qldb_session()
        logger.info('Listing table names ')
        for table in qldb_session.list_tables():
            logger.info(table)
    except ClientError:
        logger.exception('Unable to create session.')
```

> **Note**
>
> - Pour exécuter des transactions de données sur votre registre, vous devez créer un objet pilote QLDB pour vous connecter à un registre spécifique. Il s'agit d'un objet client différent de`qldb_client` celui que vous avez utilisé à l'étape précédente pour créer le registre. Ce client précédent est uniquement utilisé pour exécuter les opérations de l'API de gestion répertoriées dans leRéférence d'API Amazon QLDB.
> - Tout d'abord, créez un objet pilote QLDB groupé. Vous devez spécifier un nom de registre lorsque vous créez ce pilote.
> - Vous pouvez ensuite créer des sessions à partir de cet objet pilote groupé.

2. Pour exécuter le programme, saisissez la commande suivante.

```
python connect_to_ledger.py
```

Pour créer des tables dans le`vehicle-registration` grand livre, passez àÉtape 3 : Création de tables, d'index et d'exemples de données.

Étape 3 : Création de tables, d'index et d'exemples de données

Lorsque votre registre Amazon QLDB est actif et accepte les connexions, vous pouvez commencer à créer des tableaux contenant des données sur les véhicules, leurs propriétaires et leurs informations d'immatriculation. Après avoir créé les tables et les index, vous pouvez les charger avec des données.

Dans cette étape, vous allez créer quatre tables dans le`vehicle-registration` registre :

- `VehicleRegistration`
- `Vehicle`
- `Person`
- `DriversLicense`

Vous créez également les index suivants.

Nom de la table	Champ
`VehicleRegistration`	`VIN`
`VehicleRegistration`	`LicensePlateNumber`
`Vehicle`	`VIN`
`Person`	`GovId`
`DriversLicense`	`LicenseNumber`
`DriversLicense`	`PersonId`

Lorsque vous insérez des exemples de données, vous insérez d'abord des documents dans le`Person` tableau. Ensuite, vous utilisez le système attribué`id` à chaque`Person` document pour remplir les champs correspondants dans les`DriversLicense` documents`VehicleRegistration` et les documents appropriés.

> **Tip**
>
> La meilleure pratique consiste à utiliser une clé étrangère attribuée par`id` le système à un document. Bien que vous puissiez définir des champs destinés à être des identifiants uniques (par exemple, le VIN d'un véhicule), le véritable identifiant unique d'un document est le sien`id`. Ce champ est inclus dans les métadonnées du document, que vous pouvez interroger dans la vue validée (la vue définie par le système d'une table).
> Pour plus d'informations sur les vues dans QLDB, consultezConcepts de base. Pour en savoir plus sur les métadonnées, consultezInterroger les métadonnées d'un document.

Pour créer des tables et index

1. Utilisez le programme suivant (`create_table.py`) pour créer les tables mentionnées précédemment.

   3.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
```

```
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.constants import Constants
from pyqldbsamples.connect_to_ledger import create_qldb_driver

logger = getLogger(__name__)
basicConfig(level=INFO)


def create_table(driver, table_name):
    """
    Create a table with the specified name.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type table_name: str
    :param table_name: Name of the table to create.

    :rtype: int
    :return: The number of changes to the database.
    """
    logger.info("Creating the '{}' table...".format(table_name))
    statement = 'CREATE TABLE {}'.format(table_name)
    cursor = driver.execute_lambda(lambda executor:
 executor.execute_statement(statement))
    logger.info('{} table created successfully.'.format(table_name))
    return len(list(cursor))


def main(ledger_name=Constants.LEDGER_NAME):
    """
    Create registrations, vehicles, owners, and licenses tables.
    """
    try:
        with create_qldb_driver(ledger_name) as driver:
            create_table(driver, Constants.DRIVERS_LICENSE_TABLE_NAME)
```

```
            create_table(driver, Constants.PERSON_TABLE_NAME)
            create_table(driver, Constants.VEHICLE_TABLE_NAME)
            create_table(driver, Constants.VEHICLE_REGISTRATION_TABLE_NAME)
            logger.info('Tables created successfully.')
    except Exception as e:
        logger.exception('Errors creating tables.')
        raise e


if __name__ == '__main__':
    main()
```

2.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.constants import Constants
from pyqldbsamples.connect_to_ledger import create_qldb_session
```

```
logger = getLogger(__name__)
basicConfig(level=INFO)


def create_table(transaction_executor, table_name):
    """
    Create a table with the specified name using an Executor object.

    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.

    :type table_name: str
    :param table_name: Name of the table to create.

    :rtype: int
    :return: The number of changes to the database.
    """
    logger.info("Creating the '{}' table...".format(table_name))
    statement = 'CREATE TABLE {}'.format(table_name)
    cursor = transaction_executor.execute_statement(statement)
    logger.info('{} table created successfully.'.format(table_name))
    return len(list(cursor))


if __name__ == '__main__':
    """
    Create registrations, vehicles, owners, and licenses tables in a single
 transaction.
    """
    try:
        with create_qldb_session() as session:
            session.execute_lambda(lambda x: create_table(x,
 Constants.DRIVERS_LICENSE_TABLE_NAME) and
                                   create_table(x, Constants.PERSON_TABLE_NAME)
 and
                                   create_table(x, Constants.VEHICLE_TABLE_NAME)
 and
                                   create_table(x,
 Constants.VEHICLE_REGISTRATION_TABLE_NAME),
                                   lambda retry_attempt: logger.info('Retrying
 due to OCC conflict...'))
            logger.info('Tables created successfully.')
```

```
    except Exception:
        logger.exception('Errors creating tables.')
```

> **Note**
>
> Ce programme explique comment utiliser la`execute_lambda` fonction. Dans cet exemple, vous exécutez plusieurs instructions`CREATE TABLE` PartiQL avec une seule expression lambda.
> Cette fonction d'exécution démarre implicitement une transaction, exécute toutes les instructions du lambda, puis valide automatiquement la transaction.

2. Pour exécuter le programme, saisissez la commande suivante.

```
python create_table.py
```

3. Utilisez le programme suivant (`create_index.py`) pour créer des index sur les tables, comme décrit précédemment.

   3.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
```

```
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.constants import Constants
from pyqldbsamples.connect_to_ledger import create_qldb_driver

logger = getLogger(__name__)
basicConfig(level=INFO)


def create_index(driver, table_name, index_attribute):
    """
    Create an index for a particular table.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type table_name: str
    :param table_name: Name of the table to add indexes for.

    :type index_attribute: str
    :param index_attribute: Index to create on a single attribute.

    :rtype: int
    :return: The number of changes to the database.
    """
    logger.info("Creating index on '{}'...".format(index_attribute))
    statement = 'CREATE INDEX on {} ({})'.format(table_name, index_attribute)
    cursor = driver.execute_lambda(lambda executor:
 executor.execute_statement(statement))
    return len(list(cursor))


def main(ledger_name=Constants.LEDGER_NAME):
    """
    Create indexes on tables in a particular ledger.
    """
    logger.info('Creating indexes on all tables...')
    try:
        with create_qldb_driver(ledger_name) as driver:
```

```
            create_index(driver, Constants.PERSON_TABLE_NAME,
 Constants.GOV_ID_INDEX_NAME)
            create_index(driver, Constants.VEHICLE_TABLE_NAME,
 Constants.VEHICLE_VIN_INDEX_NAME)
            create_index(driver, Constants.VEHICLE_REGISTRATION_TABLE_NAME,
 Constants.LICENSE_PLATE_NUMBER_INDEX_NAME)
            create_index(driver, Constants.VEHICLE_REGISTRATION_TABLE_NAME,
 Constants.VEHICLE_VIN_INDEX_NAME)
            create_index(driver, Constants.DRIVERS_LICENSE_TABLE_NAME,
 Constants.PERSON_ID_INDEX_NAME)
            create_index(driver, Constants.DRIVERS_LICENSE_TABLE_NAME,
 Constants.LICENSE_NUMBER_INDEX_NAME)
            logger.info('Indexes created successfully.')
    except Exception as e:
        logger.exception('Unable to create indexes.')
        raise e


if __name__ == '__main__':
    main()
```

2.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
```

```
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.constants import Constants
from pyqldbsamples.connect_to_ledger import create_qldb_session

logger = getLogger(__name__)
basicConfig(level=INFO)


def create_index(transaction_executor, table_name, index_attribute):
    """
    Create an index for a particular table.

    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.

    :type table_name: str
    :param table_name: Name of the table to add indexes for.

    :type index_attribute: str
    :param index_attribute: Index to create on a single attribute.

    :rtype: int
    :return: The number of changes to the database.
    """
    logger.info("Creating index on '{}'...".format(index_attribute))
    statement = 'CREATE INDEX on {} ({})'.format(table_name, index_attribute)
    cursor = transaction_executor.execute_statement(statement)
    return len(list(cursor))


if __name__ == '__main__':
    """
    Create indexes on tables in a particular ledger.
    """
    logger.info('Creating indexes on all tables in a single transaction...')
    try:
        with create_qldb_session() as session:
```

```
            session.execute_lambda(lambda x: create_index(x,
Constants.PERSON_TABLE_NAME,

Constants.GOV_ID_INDEX_NAME)
                                   and create_index(x,
Constants.VEHICLE_TABLE_NAME,

Constants.VEHICLE_VIN_INDEX_NAME)
                                   and create_index(x,
Constants.VEHICLE_REGISTRATION_TABLE_NAME,

Constants.LICENSE_PLATE_NUMBER_INDEX_NAME)
                                   and create_index(x,
Constants.VEHICLE_REGISTRATION_TABLE_NAME,

Constants.VEHICLE_VIN_INDEX_NAME)
                                   and create_index(x,
Constants.DRIVERS_LICENSE_TABLE_NAME,

Constants.PERSON_ID_INDEX_NAME)
                                   and create_index(x,
Constants.DRIVERS_LICENSE_TABLE_NAME,

Constants.LICENSE_NUMBER_INDEX_NAME),
                                   lambda retry_attempt: logger.info('Retrying
due to OCC conflict...'))
            logger.info('Indexes created successfully.')
    except Exception:
        logger.exception('Unable to create indexes.')
```

4. Pour exécuter le programme, saisissez la commande suivante.

```
python create_index.py
```

Pour charger des données dans les tables

1. Consultez le fichier suivant (`sample_data.py`), qui représente les exemples de données que vous insérez dans les`vehicle-registration` tableaux. Ce fichier est également importé depuis le`amazon.ion` package pour fournir des fonctions d'assistance permettant de convertir, d'analyser et d'imprimer les données Amazon Ion.

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
from datetime import datetime
from decimal import Decimal
from logging import basicConfig, getLogger, INFO

from amazon.ion.simple_types import IonPyBool, IonPyBytes, IonPyDecimal, IonPyDict,
 IonPyFloat, IonPyInt, IonPyList, \
    IonPyNull, IonPySymbol, IonPyText, IonPyTimestamp
from amazon.ion.simpleion import dumps, loads

logger = getLogger(__name__)
basicConfig(level=INFO)
IonValue = (IonPyBool, IonPyBytes, IonPyDecimal, IonPyDict, IonPyFloat, IonPyInt,
 IonPyList, IonPyNull, IonPySymbol,
            IonPyText, IonPyTimestamp)


class SampleData:
    """
    Sample domain objects for use throughout this tutorial.
    """
    DRIVERS_LICENSE = [
        {
```

```
            'PersonId': '',
            'LicenseNumber': 'LEWISR261LL',
            'LicenseType': 'Learner',
            'ValidFromDate': datetime(2016, 12, 20),
            'ValidToDate': datetime(2020, 11, 15)
        },
        {
            'PersonId': '',
            'LicenseNumber': 'LOGANB486CG',
            'LicenseType': 'Probationary',
            'ValidFromDate': datetime(2016, 4, 6),
            'ValidToDate': datetime(2020, 11, 15)
        },
        {
            'PersonId': '',
            'LicenseNumber': '744 849 301',
            'LicenseType': 'Full',
            'ValidFromDate': datetime(2017, 12, 6),
            'ValidToDate': datetime(2022, 10, 15)
        },
        {
            'PersonId': '',
            'LicenseNumber': 'P626-168-229-765',
            'LicenseType': 'Learner',
            'ValidFromDate': datetime(2017, 8, 16),
            'ValidToDate': datetime(2021, 11, 15)
        },
        {
            'PersonId': '',
            'LicenseNumber': 'S152-780-97-415-0',
            'LicenseType': 'Probationary',
            'ValidFromDate': datetime(2015, 8, 15),
            'ValidToDate': datetime(2021, 8, 21)
        }
    ]
    PERSON = [
        {
            'FirstName': 'Raul',
            'LastName': 'Lewis',
            'Address': '1719 University Street, Seattle, WA, 98109',
            'DOB': datetime(1963, 8, 19),
            'GovId': 'LEWISR261LL',
            'GovIdType': 'Driver License'
        },
```

```
        {
            'FirstName': 'Brent',
            'LastName': 'Logan',
            'DOB': datetime(1967, 7, 3),
            'Address': '43 Stockert Hollow Road, Everett, WA, 98203',
            'GovId': 'LOGANB486CG',
            'GovIdType': 'Driver License'
        },
        {
            'FirstName': 'Alexis',
            'LastName': 'Pena',
            'DOB': datetime(1974, 2, 10),
            'Address': '4058 Melrose Street, Spokane Valley, WA, 99206',
            'GovId': '744 849 301',
            'GovIdType': 'SSN'
        },
        {
            'FirstName': 'Melvin',
            'LastName': 'Parker',
            'DOB': datetime(1976, 5, 22),
            'Address': '4362 Ryder Avenue, Seattle, WA, 98101',
            'GovId': 'P626-168-229-765',
            'GovIdType': 'Passport'
        },
        {
            'FirstName': 'Salvatore',
            'LastName': 'Spencer',
            'DOB': datetime(1997, 11, 15),
            'Address': '4450 Honeysuckle Lane, Seattle, WA, 98101',
            'GovId': 'S152-780-97-415-0',
            'GovIdType': 'Passport'
        }
    ]
    VEHICLE = [
        {
            'VIN': '1N4AL11D75C109151',
            'Type': 'Sedan',
            'Year': 2011,
            'Make': 'Audi',
            'Model': 'A5',
            'Color': 'Silver'
        },
        {
            'VIN': 'KM8SRDHF6EU074761',
```

```
            'Type': 'Sedan',
            'Year': 2015,
            'Make': 'Tesla',
            'Model': 'Model S',
            'Color': 'Blue'
        },
        {
            'VIN': '3HGGK5G53FM761765',
            'Type': 'Motorcycle',
            'Year': 2011,
            'Make': 'Ducati',
            'Model': 'Monster 1200',
            'Color': 'Yellow'
        },
        {
            'VIN': '1HVBBAANXWH544237',
            'Type': 'Semi',
            'Year': 2009,
            'Make': 'Ford',
            'Model': 'F 150',
            'Color': 'Black'
        },
        {
            'VIN': '1C4RJFAG0FC625797',
            'Type': 'Sedan',
            'Year': 2019,
            'Make': 'Mercedes',
            'Model': 'CLK 350',
            'Color': 'White'
        }
    ]
    VEHICLE_REGISTRATION = [
        {
            'VIN': '1N4AL11D75C109151',
            'LicensePlateNumber': 'LEWISR261LL',
            'State': 'WA',
            'City': 'Seattle',
            'ValidFromDate': datetime(2017, 8, 21),
            'ValidToDate': datetime(2020, 5, 11),
            'PendingPenaltyTicketAmount': Decimal('90.25'),
            'Owners': {
                'PrimaryOwner': {'PersonId': ''},
                'SecondaryOwners': []
            }
```

```
        },
        {
            'VIN': 'KM8SRDHF6EU074761',
            'LicensePlateNumber': 'CA762X',
            'State': 'WA',
            'City': 'Kent',
            'PendingPenaltyTicketAmount': Decimal('130.75'),
            'ValidFromDate': datetime(2017, 9, 14),
            'ValidToDate': datetime(2020, 6, 25),
            'Owners': {
                'PrimaryOwner': {'PersonId': ''},
                'SecondaryOwners': []
            }
        },
        {
            'VIN': '3HGGK5G53FM761765',
            'LicensePlateNumber': 'CD820Z',
            'State': 'WA',
            'City': 'Everett',
            'PendingPenaltyTicketAmount': Decimal('442.30'),
            'ValidFromDate': datetime(2011, 3, 17),
            'ValidToDate': datetime(2021, 3, 24),
            'Owners': {
                'PrimaryOwner': {'PersonId': ''},
                'SecondaryOwners': []
            }
        },
        {
            'VIN': '1HVBBAANXWH544237',
            'LicensePlateNumber': 'LS477D',
            'State': 'WA',
            'City': 'Tacoma',
            'PendingPenaltyTicketAmount': Decimal('42.20'),
            'ValidFromDate': datetime(2011, 10, 26),
            'ValidToDate': datetime(2023, 9, 25),
            'Owners': {
                'PrimaryOwner': {'PersonId': ''},
                'SecondaryOwners': []
            }
        },
        {
            'VIN': '1C4RJFAG0FC625797',
            'LicensePlateNumber': 'TH393F',
            'State': 'WA',
```

```
            'City': 'Olympia',
            'PendingPenaltyTicketAmount': Decimal('30.45'),
            'ValidFromDate': datetime(2013, 9, 2),
            'ValidToDate': datetime(2024, 3, 19),
            'Owners': {
                'PrimaryOwner': {'PersonId': ''},
                'SecondaryOwners': []
            }
        }
    ]


def convert_object_to_ion(py_object):
    """
    Convert a Python object into an Ion object.

    :type py_object: object
    :param py_object: The object to convert.

    :rtype: :py:class:`amazon.ion.simple_types.IonPyValue`
    :return: The converted Ion object.
    """
    ion_object = loads(dumps(py_object))
    return ion_object


def to_ion_struct(key, value):
    """
    Convert the given key and value into an Ion struct.

    :type key: str
    :param key: The key which serves as an unique identifier.

    :type value: str
    :param value: The value associated with a given key.

    :rtype: :py:class:`amazon.ion.simple_types.IonPyDict`
    :return: The Ion dictionary object.
    """
    ion_struct = dict()
    ion_struct[key] = value
    return loads(str(ion_struct))
```

```
def get_document_ids(transaction_executor, table_name, field, value):
    """
    Gets the document IDs from the given table.

    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.

    :type table_name: str
    :param table_name: The table name to query.

    :type field: str
    :param field: A field to query.

    :type value: str
    :param value: The key of the given field.

    :rtype: list
    :return: A list of document IDs.
    """
    query = "SELECT id FROM {} AS t BY id WHERE t.{} = ?".format(table_name, field)
    cursor = transaction_executor.execute_statement(query,
 convert_object_to_ion(value))
    return list(map(lambda table: table.get('id'), cursor))


def get_document_ids_from_dml_results(result):
    """
    Return a list of modified document IDs as strings from DML results.

    :type result: :py:class:`pyqldb.cursor.buffered_cursor.BufferedCursor`
    :param: result: The result set from DML operation.

    :rtype: list
    :return: List of document IDs.
    """
    ret_val = list(map(lambda x: x.get('documentId'), result))
    return ret_val


def print_result(cursor):
    """
    Pretty print the result set. Returns the number of documents in the result set.
```

```
    :type cursor: :py:class:`pyqldb.cursor.stream_cursor.StreamCursor`/
                  :py:class:`pyqldb.cursor.buffered_cursor.BufferedCursor`
    :param cursor: An instance of the StreamCursor or BufferedCursor class.

    :rtype: int
    :return: Number of documents in the result set.
    """
    result_counter = 0
    for row in cursor:
        # Each row would be in Ion format.
        print_ion(row)
        result_counter += 1
    return result_counter


def print_ion(ion_value):
    """
    Pretty print an Ion Value.

    :type ion_value: :py:class:`amazon.ion.simple_types.IonPySymbol`
    :param ion_value: Any Ion Value to be pretty printed.
    """
    logger.info(dumps(ion_value, binary=False, indent='  ',
 omit_version_marker=True))
```

> **Note**
>
> La `get_document_ids` fonction exécute une requête qui renvoie les identifiants de document attribués par le système à partir d'une table. Pour en savoir plus, veuillez consulter la section Utilisation de la clause BY pour interroger l'ID du document.

2. Utilisez le programme suivant (`insert_document.py`) pour insérer les exemples de données dans vos tableaux.

   3.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
```

```
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.constants import Constants
from pyqldbsamples.model.sample_data import convert_object_to_ion, SampleData,
 get_document_ids_from_dml_results
from pyqldbsamples.connect_to_ledger import create_qldb_driver

logger = getLogger(__name__)
basicConfig(level=INFO)


def update_person_id(document_ids):
    """
    Update the PersonId value for DriversLicense records and the PrimaryOwner
 value for VehicleRegistration records.

    :type document_ids: list
    :param document_ids: List of document IDs.

    :rtype: list
    :return: Lists of updated DriversLicense records and updated
 VehicleRegistration records.
    """
    new_drivers_licenses = SampleData.DRIVERS_LICENSE.copy()
```

```
    new_vehicle_registrations = SampleData.VEHICLE_REGISTRATION.copy()
    for i in range(len(SampleData.PERSON)):
        drivers_license = new_drivers_licenses[i]
        registration = new_vehicle_registrations[i]
        drivers_license.update({'PersonId': str(document_ids[i])})
        registration['Owners']['PrimaryOwner'].update({'PersonId':
 str(document_ids[i])})
    return new_drivers_licenses, new_vehicle_registrations


def insert_documents(driver, table_name, documents):
    """
    Insert the given list of documents into a table in a single transaction.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type table_name: str
    :param table_name: Name of the table to insert documents into.

    :type documents: list
    :param documents: List of documents to insert.

    :rtype: list
    :return: List of documents IDs for the newly inserted documents.
    """
    logger.info('Inserting some documents in the {}
 table...'.format(table_name))
    statement = 'INSERT INTO {} ?'.format(table_name)
    cursor = driver.execute_lambda(lambda executor:
 executor.execute_statement(statement,

 convert_object_to_ion(documents)))
    list_of_document_ids = get_document_ids_from_dml_results(cursor)

    return list_of_document_ids


def update_and_insert_documents(driver):
    """
    Handle the insertion of documents and updating PersonIds.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.
```

```
    """
    list_ids = insert_documents(driver, Constants.PERSON_TABLE_NAME,
 SampleData.PERSON)

    logger.info("Updating PersonIds for 'DriversLicense' and PrimaryOwner for
 'VehicleRegistration'...")
    new_licenses, new_registrations = update_person_id(list_ids)

    insert_documents(driver, Constants.VEHICLE_TABLE_NAME, SampleData.VEHICLE)
    insert_documents(driver, Constants.VEHICLE_REGISTRATION_TABLE_NAME,
 new_registrations)
    insert_documents(driver, Constants.DRIVERS_LICENSE_TABLE_NAME, new_licenses)


def main(ledger_name=Constants.LEDGER_NAME):
    """
    Insert documents into a table in a QLDB ledger.
    """
    try:
        with create_qldb_driver(ledger_name) as driver:
            # An INSERT statement creates the initial revision of a document
 with a version number of zero.
            # QLDB also assigns a unique document identifier in GUID format as
 part of the metadata.
            update_and_insert_documents(driver)
            logger.info('Documents inserted successfully!')
    except Exception as e:
        logger.exception('Error inserting or updating documents.')
        raise e


if __name__ == '__main__':
    main()
```

2.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
```

```
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.constants import Constants
from pyqldbsamples.model.sample_data import convert_object_to_ion, SampleData,
 get_document_ids_from_dml_results
from pyqldbsamples.connect_to_ledger import create_qldb_session

logger = getLogger(__name__)
basicConfig(level=INFO)


def update_person_id(document_ids):
    """
    Update the PersonId value for DriversLicense records and the PrimaryOwner
 value for VehicleRegistration records.

    :type document_ids: list
    :param document_ids: List of document IDs.

    :rtype: list
    :return: Lists of updated DriversLicense records and updated
 VehicleRegistration records.
    """
    new_drivers_licenses = SampleData.DRIVERS_LICENSE.copy()
    new_vehicle_registrations = SampleData.VEHICLE_REGISTRATION.copy()
    for i in range(len(SampleData.PERSON)):
```

```
        drivers_license = new_drivers_licenses[i]
        registration = new_vehicle_registrations[i]
        drivers_license.update({'PersonId': str(document_ids[i])})
        registration['Owners']['PrimaryOwner'].update({'PersonId':
 str(document_ids[i])})
    return new_drivers_licenses, new_vehicle_registrations


def insert_documents(transaction_executor, table_name, documents):
    """
    Insert the given list of documents into a table in a single transaction.

    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.

    :type table_name: str
    :param table_name: Name of the table to insert documents into.

    :type documents: list
    :param documents: List of documents to insert.

    :rtype: list
    :return: List of documents IDs for the newly inserted documents.
    """
    logger.info('Inserting some documents in the {}
 table...'.format(table_name))
    statement = 'INSERT INTO {} ?'.format(table_name)
    cursor = transaction_executor.execute_statement(statement,
 convert_object_to_ion(documents))
    list_of_document_ids = get_document_ids_from_dml_results(cursor)

    return list_of_document_ids


def update_and_insert_documents(transaction_executor):
    """
    Handle the insertion of documents and updating PersonIds all in a single
 transaction.

    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.
    """
```

```
    list_ids = insert_documents(transaction_executor,
 Constants.PERSON_TABLE_NAME, SampleData.PERSON)

    logger.info("Updating PersonIds for 'DriversLicense' and PrimaryOwner for
 'VehicleRegistration'...")
    new_licenses, new_registrations = update_person_id(list_ids)

    insert_documents(transaction_executor, Constants.VEHICLE_TABLE_NAME,
 SampleData.VEHICLE)
    insert_documents(transaction_executor,
 Constants.VEHICLE_REGISTRATION_TABLE_NAME, new_registrations)
    insert_documents(transaction_executor, Constants.DRIVERS_LICENSE_TABLE_NAME,
 new_licenses)


if __name__ == '__main__':
    """
    Insert documents into a table in a QLDB ledger.
    """
    try:
        with create_qldb_session() as session:
            # An INSERT statement creates the initial revision of a document
 with a version number of zero.
            # QLDB also assigns a unique document identifier in GUID format as
 part of the metadata.
            session.execute_lambda(lambda executor:
 update_and_insert_documents(executor),
                                   lambda retry_attempt: logger.info('Retrying
 due to OCC conflict...'))
            logger.info('Documents inserted successfully!')
    except Exception:
        logger.exception('Error inserting or updating documents.')
```

Note

- Ce programme montre comment appeler la`execute_statement` fonction avec des valeurs paramétrées. Vous pouvez transmettre des paramètres de données en plus de l'instruction PartiQL en plus de l'instruction PartiQL en plus de l'instruction PartiQL. Utilisez un point d'interrogation (?) comme espace réservé à une variable dans votre chaîne de déclaration.

- Si une`INSERT` instruction aboutit, elle renvoie la valeur`id` de chaque document inséré.

3. Pour exécuter le programme, saisissez la commande suivante.

```
python insert_document.py
```

Vous pouvez ensuite utiliser`SELECT` des instructions pour lire les données des tables du`vehicle-registration` registre. Passez à Étape 4 : interroger les tables d'un registre.

Étape 4 : interroger les tables d'un registre

Après avoir créé des tables dans un registre Amazon QLDB et les avoir chargées avec des données, vous pouvez exécuter des requêtes pour vérifier les données d'immatriculation du véhicule que vous venez d'insérer. QLDB utilise PartiQL comme langage de requête et Amazon Ion comme modèle de données orienté document.

PartiQL est un langage de requête open source compatible SQL qui a été étendu pour fonctionner avec Ion. Avec PartiQL, vous pouvez insérer, interroger et gérer vos données à l'aide d'opérateurs SQL familiers. Amazon Ion est un sur-ensemble de JSON. Ion est un format de données open source basé sur des documents qui vous permet de stocker et de traiter des données structurées, semi-structurées et imbriquées.

Au cours de cette étape, vous utilisez des`SELECT` instructions pour lire les données des tables du`vehicle-registration` registre.

Warning

Lorsque vous exécutez une requête dans QLDB sans recherche indexée, elle appelle une analyse complète de la table. PartiQL prend en charge de telles requêtes car il est compatible avec SQL. Toutefois, n'exécutez pas d'analyses de tables pour des cas d'utilisation en production dans QLDB. Les analyses de tables peuvent entraîner des problèmes de performances sur des tables de grande taille, notamment des conflits de simultanéité et des délais de transaction.

Pour éviter de scanner des tables, vous devez exécuter des instructions avec une clause de`WHERE` prédicat à l'aide d'un opérateur d'égalité sur un champ indexé ou un identifiant de document, par exemple,`WHERE indexedField = 123` ou`WHERE indexedField IN`

`(456, 789)`. Pour plus d'informations, veuillez consulter Optimisation des performances des données.

Pour interroger les tables

1. Utilisez le programme suivant (`find_vehicles.py`) pour rechercher tous les véhicules enregistrés par une personne dans votre registre.

   3.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.model.sample_data import get_document_ids, print_result,
 SampleData
from pyqldbsamples.constants import Constants
from pyqldbsamples.connect_to_ledger import create_qldb_driver
```

```
logger = getLogger(__name__)
basicConfig(level=INFO)


def find_vehicles_for_owner(driver, gov_id):
    """
    Find vehicles registered under a driver using their government ID.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type gov_id: str
    :param gov_id: The owner's government ID.
    """
    document_ids = driver.execute_lambda(lambda executor:
 get_document_ids(executor, Constants.PERSON_TABLE_NAME,

 'GovId', gov_id))

    query = "SELECT Vehicle FROM Vehicle INNER JOIN VehicleRegistration AS r " \
            "ON Vehicle.VIN = r.VIN WHERE r.Owners.PrimaryOwner.PersonId = ?"

    for ids in document_ids:
        cursor = driver.execute_lambda(lambda executor:
 executor.execute_statement(query, ids))
        logger.info('List of Vehicles for owner with GovId:
 {}...'.format(gov_id))
        print_result(cursor)


def main(ledger_name=Constants.LEDGER_NAME):
    """
    Find all vehicles registered under a person.
    """
    try:
        with create_qldb_driver(ledger_name) as driver:
            # Find all vehicles registered under a person.
            gov_id = SampleData.PERSON[0]['GovId']
            find_vehicles_for_owner(driver, gov_id)
    except Exception as e:
        logger.exception('Error getting vehicles for owner.')
        raise e
```

```
if __name__ == '__main__':
    main()
```

2.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.model.sample_data import get_document_ids, print_result,
 SampleData
from pyqldbsamples.constants import Constants
from pyqldbsamples.connect_to_ledger import create_qldb_session

logger = getLogger(__name__)
basicConfig(level=INFO)


def find_vehicles_for_owner(transaction_executor, gov_id):
    """
```

```
    Find vehicles registered under a driver using their government ID.

    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
statements within a transaction.

    :type gov_id: str
    :param gov_id: The owner's government ID.
    """
    document_ids = get_document_ids(transaction_executor,
Constants.PERSON_TABLE_NAME, 'GovId', gov_id)

    query = "SELECT Vehicle FROM Vehicle INNER JOIN VehicleRegistration AS r " \
            "ON Vehicle.VIN = r.VIN WHERE r.Owners.PrimaryOwner.PersonId = ?"

    for ids in document_ids:
        cursor = transaction_executor.execute_statement(query, ids)
        logger.info('List of Vehicles for owner with GovId:
{}...'.format(gov_id))
        print_result(cursor)


if __name__ == '__main__':
    """
    Find all vehicles registered under a person.
    """
    try:
        with create_qldb_session() as session:
            # Find all vehicles registered under a person.
            gov_id = SampleData.PERSON[0]['GovId']
            session.execute_lambda(lambda executor:
find_vehicles_for_owner(executor, gov_id),
                                   lambda retry_attempt: logger.info('Retrying
due to OCC conflict...'))
    except Exception:
        logger.exception('Error getting vehicles for owner.')
```

> **Note**
>
> Tout d'abord, ce programme interroge la`Person` table contenant le document`GovId` `LEWISR261LL` pour obtenir son champ `id` métadonnées.

Il utilise ensuite ce document`id` comme clé étrangère pour interroger la`VehicleRegistration` table`PrimaryOwner.PersonId`. Il se joint également`VehicleRegistration` à la`Vehicle` table sur le`VIN` terrain.

2. Pour exécuter le programme, saisissez la commande suivante.

```
python find_vehicles.py
```

Pour en savoir plus sur la modification de documents dans les tableaux du`vehicle-registration` grand livre, reportez-vous à la sectionÉtape 5 : Modifier des documents dans un registre.

Étape 5 : Modifier des documents dans un registre

Maintenant que vous avez des données sur lesquelles travailler, vous pouvez commencer à apporter des modifications aux documents du`vehicle-registration` registre dans Amazon QLDB. Dans cette étape, les exemples de code suivants montrent comment exécuter des instructions de définition de données (DML). Ces déclarations mettent à jour le nom du propriétaire principal d'un véhicule et ajoutent un propriétaire secondaire à un autre véhicule.

Pour modifier des documents

1. Utilisez le programme suivant (`transfer_vehicle_ownership.py`) pour indiquer au propriétaire principal du véhicule le VIN`1N4AL11D75C109151` dans votre registre.

   3.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
```

```
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.add_secondary_owner import get_document_ids, print_result,
 SampleData
from pyqldbsamples.constants import Constants
from pyqldbsamples.model.sample_data import convert_object_to_ion
from pyqldbsamples.connect_to_ledger import create_qldb_driver

logger = getLogger(__name__)
basicConfig(level=INFO)


def find_person_from_document_id(transaction_executor, document_id):
    """
    Query a driver's information using the given ID.

    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.

    :type document_id: :py:class:`amazon.ion.simple_types.IonPyText`
    :param document_id: The document ID required to query for the person.

    :rtype: :py:class:`amazon.ion.simple_types.IonPyDict`
    :return: The resulting document from the query.
    """
    query = 'SELECT p.* FROM Person AS p BY pid WHERE pid = ?'
    cursor = transaction_executor.execute_statement(query, document_id)
    return next(cursor)


def find_primary_owner_for_vehicle(driver, vin):
```

```
    """
    Find the primary owner of a vehicle given its VIN.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type vin: str
    :param vin: The VIN to find primary owner for.

    :rtype: :py:class:`amazon.ion.simple_types.IonPyDict`
    :return: The resulting document from the query.
    """
    logger.info('Finding primary owner for vehicle with VIN: {}.'.format(vin))
    query = "SELECT Owners.PrimaryOwner.PersonId FROM VehicleRegistration AS v
 WHERE v.VIN = ?"
    cursor = driver.execute_lambda(lambda executor:
 executor.execute_statement(query, convert_object_to_ion(vin)))
    try:
        return driver.execute_lambda(lambda executor:
 find_person_from_document_id(executor,

   next(cursor).get('PersonId')))
    except StopIteration:
        logger.error('No primary owner registered for this vehicle.')
        return None


def update_vehicle_registration(driver, vin, document_id):
    """
    Update the primary owner for a vehicle using the given VIN.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type vin: str
    :param vin: The VIN for the vehicle to operate on.

    :type document_id: :py:class:`amazon.ion.simple_types.IonPyText`
    :param document_id: New PersonId for the primary owner.

    :raises RuntimeError: If no vehicle registration was found using the given
 document ID and VIN.
    """
```

```
    logger.info('Updating the primary owner for vehicle with Vin:
 {}...'.format(vin))
    statement = "UPDATE VehicleRegistration AS r SET
 r.Owners.PrimaryOwner.PersonId = ? WHERE r.VIN = ?"
    cursor = driver.execute_lambda(lambda executor:
 executor.execute_statement(statement, document_id,

 convert_object_to_ion(vin)))
    try:
        print_result(cursor)
        logger.info('Successfully transferred vehicle with VIN: {} to new
 owner.'.format(vin))
    except StopIteration:
        raise RuntimeError('Unable to transfer vehicle, could not find
 registration.')


def validate_and_update_registration(driver, vin, current_owner, new_owner):
    """
    Validate the current owner of the given vehicle and transfer its ownership
 to a new owner.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type vin: str
    :param vin: The VIN of the vehicle to transfer ownership of.

    :type current_owner: str
    :param current_owner: The GovId of the current owner of the vehicle.

    :type new_owner: str
    :param new_owner: The GovId of the new owner of the vehicle.

    :raises RuntimeError: If unable to verify primary owner.
    """
    primary_owner = find_primary_owner_for_vehicle(driver, vin)
    if primary_owner is None or primary_owner['GovId'] != current_owner:
        raise RuntimeError('Incorrect primary owner identified for vehicle,
 unable to transfer.')

    document_ids = driver.execute_lambda(lambda executor:
 get_document_ids(executor, Constants.PERSON_TABLE_NAME,
```

```
 'GovId', new_owner))
    update_vehicle_registration(driver, vin, document_ids[0])


def main(ledger_name=Constants.LEDGER_NAME):
    """
    Find primary owner for a particular vehicle's VIN.
    Transfer to another primary owner for a particular vehicle's VIN.
    """
    vehicle_vin = SampleData.VEHICLE[0]['VIN']
    previous_owner = SampleData.PERSON[0]['GovId']
    new_owner = SampleData.PERSON[1]['GovId']

    try:
        with create_qldb_driver(ledger_name) as driver:
            validate_and_update_registration(driver, vehicle_vin,
 previous_owner, new_owner)
    except Exception as e:
        logger.exception('Error updating VehicleRegistration.')
        raise e


if __name__ == '__main__':
    main()
```

2.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
```

```
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.add_secondary_owner import get_document_ids, print_result,
 SampleData
from pyqldbsamples.constants import Constants
from pyqldbsamples.model.sample_data import convert_object_to_ion
from pyqldbsamples.connect_to_ledger import create_qldb_session

logger = getLogger(__name__)
basicConfig(level=INFO)


def find_person_from_document_id(transaction_executor, document_id):
    """
    Query a driver's information using the given ID.

    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.

    :type document_id: :py:class:`amazon.ion.simple_types.IonPyText`
    :param document_id: The document ID required to query for the person.

    :rtype: :py:class:`amazon.ion.simple_types.IonPyDict`
    :return: The resulting document from the query.
    """
    query = 'SELECT p.* FROM Person AS p BY pid WHERE pid = ?'
    cursor = transaction_executor.execute_statement(query, document_id)
    return next(cursor)


def find_primary_owner_for_vehicle(transaction_executor, vin):
    """
    Find the primary owner of a vehicle given its VIN.
```

```
    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.

    :type vin: str
    :param vin: The VIN to find primary owner for.

    :rtype: :py:class:`amazon.ion.simple_types.IonPyDict`
    :return: The resulting document from the query.
    """
    logger.info('Finding primary owner for vehicle with VIN: {}.'.format(vin))
    query = "SELECT Owners.PrimaryOwner.PersonId FROM VehicleRegistration AS v
 WHERE v.VIN = ?"
    cursor = transaction_executor.execute_statement(query,
 convert_object_to_ion(vin))
    try:
        return find_person_from_document_id(transaction_executor,
 next(cursor).get('PersonId'))
    except StopIteration:
        logger.error('No primary owner registered for this vehicle.')
        return None


def update_vehicle_registration(transaction_executor, vin, document_id):
    """
    Update the primary owner for a vehicle using the given VIN.

    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.

    :type vin: str
    :param vin: The VIN for the vehicle to operate on.

    :type document_id: :py:class:`amazon.ion.simple_types.IonPyText`
    :param document_id: New PersonId for the primary owner.

    :raises RuntimeError: If no vehicle registration was found using the given
 document ID and VIN.
    """
    logger.info('Updating the primary owner for vehicle with Vin:
 {}...'.format(vin))
```

```
    statement = "UPDATE VehicleRegistration AS r SET
 r.Owners.PrimaryOwner.PersonId = ? WHERE r.VIN = ?"
    cursor = transaction_executor.execute_statement(statement, document_id,
 convert_object_to_ion(vin))
    try:
        print_result(cursor)
        logger.info('Successfully transferred vehicle with VIN: {} to new
 owner.'.format(vin))
    except StopIteration:
        raise RuntimeError('Unable to transfer vehicle, could not find
 registration.')


def validate_and_update_registration(transaction_executor, vin, current_owner,
 new_owner):
    """
    Validate the current owner of the given vehicle and transfer its ownership
 to a new owner in a single transaction.

    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.

    :type vin: str
    :param vin: The VIN of the vehicle to transfer ownership of.

    :type current_owner: str
    :param current_owner: The GovId of the current owner of the vehicle.

    :type new_owner: str
    :param new_owner: The GovId of the new owner of the vehicle.

    :raises RuntimeError: If unable to verify primary owner.
    """
    primary_owner = find_primary_owner_for_vehicle(transaction_executor, vin)
    if primary_owner is None or primary_owner['GovId'] != current_owner:
        raise RuntimeError('Incorrect primary owner identified for vehicle,
 unable to transfer.')

    document_id = next(get_document_ids(transaction_executor,
 Constants.PERSON_TABLE_NAME, 'GovId', new_owner))

    update_vehicle_registration(transaction_executor, vin, document_id)
```

```
if __name__ == '__main__':
    """
    Find primary owner for a particular vehicle's VIN.
    Transfer to another primary owner for a particular vehicle's VIN.
    """
    vehicle_vin = SampleData.VEHICLE[0]['VIN']
    previous_owner = SampleData.PERSON[0]['GovId']
    new_owner = SampleData.PERSON[1]['GovId']

    try:
        with create_qldb_session() as session:
            session.execute_lambda(lambda executor:
 validate_and_update_registration(executor, vehicle_vin,

     previous_owner, new_owner),
                                   retry_indicator=lambda retry_attempt:
 logger.info('Retrying due to OCC conflict...'))
    except Exception:
        logger.exception('Error updating VehicleRegistration.')
```

2. Pour exécuter le programme, saisissez la commande suivante.

```
python transfer_vehicle_ownership.py
```

3. Utilisez le programme suivant (`add_secondary_owner.py`) pour ajouter un propriétaire secondaire au véhicule avec le VIN`KM8SRDHF6EU074761` dans votre registre.

   3.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
```

```
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.model.sample_data import to_ion_struct, get_document_ids,
 print_result, SampleData, \
    convert_object_to_ion
from pyqldbsamples.constants import Constants
from pyqldbsamples.connect_to_ledger import create_qldb_driver

logger = getLogger(__name__)
basicConfig(level=INFO)


def get_document_id_by_gov_id(driver, government_id):
    """
    Find a driver's person ID using the given government ID.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type government_id: str
    :param government_id: A driver's government ID.

    :rtype: list
    :return: A list of document IDs.
    """
    logger.info("Finding secondary owner's person ID using given government ID:
 {}.".format(government_id))
    return driver.execute_lambda(lambda executor: get_document_ids(executor,
 Constants.PERSON_TABLE_NAME, 'GovId',

 government_id))
```

```
def is_secondary_owner_for_vehicle(driver, vin, secondary_owner_id):
    """
    Check whether a secondary owner has already been registered for the given
 VIN.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type vin: str
    :param vin: VIN of the vehicle to query.

    :type secondary_owner_id: str
    :param secondary_owner_id: The secondary owner's person ID.

    :rtype: bool
    :return: If the driver has already been registered.
    """
    logger.info('Finding secondary owners for vehicle with VIN:
 {}...'.format(vin))
    query = 'SELECT Owners.SecondaryOwners FROM VehicleRegistration AS v WHERE
 v.VIN = ?'
    rows = driver.execute_lambda(lambda executor:
 executor.execute_statement(query, convert_object_to_ion(vin)))

    for row in rows:
        secondary_owners = row.get('SecondaryOwners')
        person_ids = map(lambda owner: owner.get('PersonId').text,
 secondary_owners)
        if secondary_owner_id in person_ids:
            return True
    return False


def add_secondary_owner_for_vin(driver, vin, parameter):
    """
    Add a secondary owner into `VehicleRegistration` table for a particular VIN.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type vin: str
    :param vin: VIN of the vehicle to add a secondary owner for.
```

```
    :type parameter: :py:class:`amazon.ion.simple_types.IonPyValue`
    :param parameter: The Ion value or Python native type that is convertible to
 Ion for filling in parameters of the
                      statement.
    """
    logger.info('Inserting secondary owner for vehicle with VIN:
 {}...'.format(vin))
    statement = "FROM VehicleRegistration AS v WHERE v.VIN = ? INSERT INTO
 v.Owners.SecondaryOwners VALUE ?"

    cursor = driver.execute_lambda(lambda executor:
 executor.execute_statement(statement, convert_object_to_ion(vin),

 parameter))
    logger.info('VehicleRegistration Document IDs which had secondary owners
 added: ')
    print_result(cursor)


def register_secondary_owner(driver, vin, gov_id):
    """
    Register a secondary owner for a vehicle if they are not already registered.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type vin: str
    :param vin: VIN of the vehicle to register a secondary owner for.

    :type gov_id: str
    :param gov_id: The government ID of the owner.
    """
    logger.info('Finding the secondary owners for vehicle with VIN:
 {}.'.format(vin))

    document_ids = get_document_id_by_gov_id(driver, gov_id)

    for document_id in document_ids:
        if is_secondary_owner_for_vehicle(driver, vin, document_id):
            logger.info('Person with ID {} has already been added as a secondary
 owner of this vehicle.'.format(gov_id))
        else:
            add_secondary_owner_for_vin(driver, vin, to_ion_struct('PersonId',
 document_id))
```

```
def main(ledger_name=Constants.LEDGER_NAME):
    """
    Finds and adds secondary owners for a vehicle.
    """
    vin = SampleData.VEHICLE[1]['VIN']
    gov_id = SampleData.PERSON[0]['GovId']
    try:
        with create_qldb_driver(ledger_name) as driver:
            register_secondary_owner(driver, vin, gov_id)
            logger.info('Secondary owners successfully updated.')
    except Exception as e:
        logger.exception('Error adding secondary owner.')
        raise e


if __name__ == '__main__':
    main()
```

2.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
```

```
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.model.sample_data import to_ion_struct, get_document_ids,
 print_result, SampleData, \
    convert_object_to_ion
from pyqldbsamples.constants import Constants
from pyqldbsamples.connect_to_ledger import create_qldb_session

logger = getLogger(__name__)
basicConfig(level=INFO)


def get_document_id_by_gov_id(transaction_executor, government_id):
    """
    Find a driver's person ID using the given government ID.
    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.
    :type government_id: str
    :param government_id: A driver's government ID.
    :rtype: list
    :return: A list of document IDs.
    """
    logger.info("Finding secondary owner's person ID using given government ID:
 {}.".format(government_id))
    return get_document_ids(transaction_executor, Constants.PERSON_TABLE_NAME,
 'GovId', government_id)


def is_secondary_owner_for_vehicle(transaction_executor, vin,
 secondary_owner_id):
    """
    Check whether a secondary owner has already been registered for the given
 VIN.
    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.
    :type vin: str
    :param vin: VIN of the vehicle to query.
    :type secondary_owner_id: str
```

```
    :param secondary_owner_id: The secondary owner's person ID.
    :rtype: bool
    :return: If the driver has already been registered.
    """
    logger.info('Finding secondary owners for vehicle with VIN:
 {}...'.format(vin))
    query = 'SELECT Owners.SecondaryOwners FROM VehicleRegistration AS v WHERE
 v.VIN = ?'
    rows = transaction_executor.execute_statement(query,
 convert_object_to_ion(vin))

    for row in rows:
        secondary_owners = row.get('SecondaryOwners')
        person_ids = map(lambda owner: owner.get('PersonId').text,
 secondary_owners)
        if secondary_owner_id in person_ids:
            return True
    return False


def add_secondary_owner_for_vin(transaction_executor, vin, parameter):
    """
    Add a secondary owner into `VehicleRegistration` table for a particular VIN.
    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.
    :type vin: str
    :param vin: VIN of the vehicle to add a secondary owner for.
    :type parameter: :py:class:`amazon.ion.simple_types.IonPyValue`
    :param parameter: The Ion value or Python native type that is convertible to
 Ion for filling in parameters of the
                      statement.
    """
    logger.info('Inserting secondary owner for vehicle with VIN:
 {}...'.format(vin))
    statement = "FROM VehicleRegistration AS v WHERE v.VIN = '{}' INSERT INTO
 v.Owners.SecondaryOwners VALUE ?"\
        .format(vin)

    cursor = transaction_executor.execute_statement(statement, parameter)
    logger.info('VehicleRegistration Document IDs which had secondary owners
 added: ')
    print_result(cursor)
```

```
def register_secondary_owner(transaction_executor, vin, gov_id):
    """
    Register a secondary owner for a vehicle if they are not already registered.
    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.
    :type vin: str
    :param vin: VIN of the vehicle to register a secondary owner for.
    :type gov_id: str
    :param gov_id: The government ID of the owner.
    """
    logger.info('Finding the secondary owners for vehicle with VIN:
 {}.'.format(vin))
    document_ids = get_document_id_by_gov_id(transaction_executor, gov_id)

    for document_id in document_ids:
        if is_secondary_owner_for_vehicle(transaction_executor, vin,
 document_id):
            logger.info('Person with ID {} has already been added as a secondary
 owner of this vehicle.'.format(gov_id))
        else:
            add_secondary_owner_for_vin(transaction_executor, vin,
 to_ion_struct('PersonId', document_id))


if __name__ == '__main__':
    """
    Finds and adds secondary owners for a vehicle.
    """
    vin = SampleData.VEHICLE[1]['VIN']
    gov_id = SampleData.PERSON[0]['GovId']
    try:
        with create_qldb_session() as session:
            session.execute_lambda(lambda executor:
 register_secondary_owner(executor, vin, gov_id),
                                   lambda retry_attempt: logger.info('Retrying
 due to OCC conflict...'))
            logger.info('Secondary owners successfully updated.')
    except Exception:
        logger.exception('Error adding secondary owner.')
```

4. Pour exécuter le programme, saisissez la commande suivante.

```
python add_secondary_owner.py
```

Pour consulter ces modifications dans levehicle-registration registre, consultezÉtape 6 : Afficher l'historique des révisions d'un document.

Étape 6 : Afficher l'historique des révisions d'un document

Après avoir modifié les données d'immatriculation d'un véhicule à l'étape précédente, vous pouvez consulter l'historique de tous ses propriétaires enregistrés et tout autre champ mis à jour. Au cours de cette étape, vous interrogez l'historique des révisions d'un document dans leVehicleRegistration tableau de votrevehicle-registration registre.

Pour consulter l'historique des révisions

1. Utilisez le programme suivant (query_history.py) pour interroger l'historique des révisions duVehicleRegistration document avec VIN1N4AL11D75C109151.

   3.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
```

```
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from datetime import datetime, timedelta
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.model.sample_data import print_result, get_document_ids,
 SampleData
from pyqldbsamples.constants import Constants
from pyqldbsamples.connect_to_ledger import create_qldb_driver

logger = getLogger(__name__)
basicConfig(level=INFO)


def format_date_time(date_time):
    """
    Format the given date time to a string.

    :type date_time: :py:class:`datetime.datetime`
    :param date_time: The date time to format.

    :rtype: str
    :return: The formatted date time.
    """
    return date_time.strftime('`%Y-%m-%dT%H:%M:%S.%fZ`')


def previous_primary_owners(driver, vin):
    """
    Find previous primary owners for the given VIN in a single transaction.
    In this example, query the `VehicleRegistration` history table to find all
 previous primary owners for a VIN.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type vin: str
    :param vin: VIN to find previous primary owners for.
    """
    person_ids = driver.execute_lambda(lambda executor:
 get_document_ids(executor,
```

```
 Constants.VEHICLE_REGISTRATION_TABLE_NAME,
                                                                        'VIN',
 vin))

    todays_date = datetime.utcnow() - timedelta(seconds=1)
    three_months_ago = todays_date - timedelta(days=90)
    query = 'SELECT data.Owners.PrimaryOwner, metadata.version FROM history({},
 {}, {}) AS h WHERE h.metadata.id = ?'.\
        format(Constants.VEHICLE_REGISTRATION_TABLE_NAME,
 format_date_time(three_months_ago),
               format_date_time(todays_date))

    for ids in person_ids:
        logger.info("Querying the 'VehicleRegistration' table's history using
 VIN: {}.".format(vin))
        cursor = driver.execute_lambda(lambda executor:
 executor.execute_statement(query, ids))
        if not (print_result(cursor)) > 0:
            logger.info('No modification history found within the given time
 frame for document ID: {}'.format(ids))


def main(ledger_name=Constants.LEDGER_NAME):
    """
    Query a table's history for a particular set of documents.
    """
    try:
        with create_qldb_driver(ledger_name) as driver:
            vin = SampleData.VEHICLE_REGISTRATION[0]['VIN']
            previous_primary_owners(driver, vin)
            logger.info('Successfully queried history.')
    except Exception as e:
        logger.exception('Unable to query history to find previous owners.')
        raise e


if __name__ == '__main__':
    main()
```

2.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from datetime import datetime, timedelta
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.model.sample_data import print_result, get_document_ids,
 SampleData
from pyqldbsamples.constants import Constants
from pyqldbsamples.connect_to_ledger import create_qldb_session

logger = getLogger(__name__)
basicConfig(level=INFO)


def format_date_time(date_time):
    """
    Format the given date time to a string.

    :type date_time: :py:class:`datetime.datetime`
    :param date_time: The date time to format.

    :rtype: str
```

```
    :return: The formatted date time.
    """
    return date_time.strftime('`%Y-%m-%dT%H:%M:%S.%fZ`')


def previous_primary_owners(transaction_executor, vin):
    """
    Find previous primary owners for the given VIN in a single transaction.
    In this example, query the `VehicleRegistration` history table to find all
 previous primary owners for a VIN.

    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.

    :type vin: str
    :param vin: VIN to find previous primary owners for.
    """
    person_ids = get_document_ids(transaction_executor,
 Constants.VEHICLE_REGISTRATION_TABLE_NAME, 'VIN', vin)

    todays_date = datetime.utcnow() - timedelta(seconds=1)
    three_months_ago = todays_date - timedelta(days=90)
    query = 'SELECT data.Owners.PrimaryOwner, metadata.version FROM history({},
 {}, {}) AS h WHERE h.metadata.id = ?'.\
        format(Constants.VEHICLE_REGISTRATION_TABLE_NAME,
 format_date_time(three_months_ago),
               format_date_time(todays_date))

    for ids in person_ids:
        logger.info("Querying the 'VehicleRegistration' table's history using
 VIN: {}.".format(vin))
        cursor = transaction_executor.execute_statement(query, ids)
        if not (print_result(cursor)) > 0:
            logger.info('No modification history found within the given time
 frame for document ID: {}'.format(ids))


if __name__ == '__main__':
    """
    Query a table's history for a particular set of documents.
    """
    try:
        with create_qldb_session() as session:
```

```
            vin = SampleData.VEHICLE_REGISTRATION[0]['VIN']
            session.execute_lambda(lambda lambda_executor:
previous_primary_owners(lambda_executor, vin),
                                   lambda retry_attempt: logger.info('Retrying
due to OCC conflict...'))
            logger.info('Successfully queried history.')
    except Exception:
        logger.exception('Unable to query history to find previous owners.')
```

> **Note**
>
> - Vous pouvez consulter l'historique des révisions d'un documentFonction historique en interrogeant la syntaxe intégrée suivante.
>
>   ```
>   SELECT * FROM history( table_name [, `start-time` [, `end-time` ] ] ) AS h
>   [ WHERE h.metadata.id = 'id' ]
>   ```
>
> - L'heure de début et l'heure de fin sont toutes deux facultatives. Ce sont des valeurs littérales d'Amazon Ion qui peuvent être signalées par des crochets inverses (`` `...` ``). Pour en savoir plus, veuillez consulter la section Interroger Ion avec PartiQL dans Amazon QLDB.
> - Il est recommandé de qualifier une requête d'historique à la fois avec une plage de dates (heure de début et heure de fin) et un identifiant de document (`metadata.id`). QLDB traite les`SELECT` requêtes dans les transactions, qui sont soumises à un délai d'expiration des transactions.
>
>   L'historique QLDB est indexé par ID de document et vous ne pouvez pas créer d'index d'historique supplémentaires pour le moment. Les requêtes d'historique qui incluent une heure de début et une heure de fin bénéficient d'une qualification par plage de dates.

2. Pour exécuter le programme, saisissez la commande suivante.

```
python query_history.py
```

Pour vérifier cryptographiquement la révision d'un document dans le`vehicle-registration` registre, passez àÉtape 7 : Vérification d'un document dans un registre.

Étape 7 : Vérification d'un document dans un registre

Avec Amazon QLDB, vous pouvez vérifier efficacement l'intégrité d'un document dans le journal de votre registre en utilisant le hachage cryptographique SHA-256. Pour en savoir plus sur le fonctionnement de la vérification et du hachage cryptographique dans QLDB, consultezVérification des données dans Amazon QLDB.

Au cours de cette étape, vous vérifiez la révision d'un document dans le`VehicleRegistration` tableau de votre`vehicle-registration` registre. Tout d'abord, vous demandez un condensé, qui est renvoyé sous forme de fichier de sortie et fait office de signature de l'historique complet des modifications de votre registre. Ensuite, vous demandez une preuve pour la révision par rapport à ce condensé. À l'aide de cette preuve, l'intégrité de votre révision est vérifiée si tous les contrôles de validation sont réussis.

Pour vérifier la révision d'un document

1. Consultez les`.py` fichiers suivants, qui représentent les objets QLDB requis pour la vérification et un module utilitaire doté de fonctions d'assistance permettant de convertir les types de réponses QLDB en chaînes.

   1. `block_address.py`

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
```

```
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.


def block_address_to_dictionary(ion_dict):
    """
    Convert a block address from IonPyDict into a dictionary.
    Shape of the dictionary must be: {'IonText': "{strandId: <"strandId">,
 sequenceNo: <sequenceNo>}"}

    :type ion_dict: :py:class:`amazon.ion.simple_types.IonPyDict`/str
    :param ion_dict: The block address value to convert.

    :rtype: dict
    :return: The converted dict.
    """
    block_address = {'IonText': {}}
    if not isinstance(ion_dict, str):
        py_dict = '{{strandId: "{}", sequenceNo:
{}}}'.format(ion_dict['strandId'], ion_dict['sequenceNo'])
        ion_dict = py_dict
    block_address['IonText'] = ion_dict
    return block_address
```

2. verifier.py

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
```

```
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from array import array
from base64 import b64encode
from functools import reduce
from hashlib import sha256
from random import randrange

from amazon.ion.simpleion import loads

HASH_LENGTH = 32
UPPER_BOUND = 8


def parse_proof(value_holder):
    """
    Parse the Proof object returned by QLDB into an iterator.

    The Proof object returned by QLDB is a dictionary like the following:
    {'IonText': '[{{<hash>}},{{<hash>}}]'}

    :type value_holder: dict
    :param value_holder: A structure containing an Ion string value.

    :rtype: :py:class:`amazon.ion.simple_types.IonPyList`
    :return: A list of hash values.
    """
    value_holder = value_holder.get('IonText')
    proof_list = loads(value_holder)
    return proof_list


def parse_block(value_holder):
    """
    Parse the Block object returned by QLDB and retrieve block hash.

    :type value_holder: dict
    :param value_holder: A structure containing an Ion string value.
```

```
    :rtype: :py:class:`amazon.ion.simple_types.IonPyBytes`
    :return: The block hash.
    """
    value_holder = value_holder.get('IonText')
    block = loads(value_holder)
    block_hash = block.get('blockHash')
    return block_hash


def flip_random_bit(original):
    """
    Flip a single random bit in the given hash value.
    This method is used to demonstrate QLDB's verification features.

    :type original: bytes
    :param original: The hash value to alter.

    :rtype: bytes
    :return: The altered hash with a single random bit changed.
    """
    assert len(original) != 0, 'Invalid bytes.'

    altered_position = randrange(len(original))
    bit_shift = randrange(UPPER_BOUND)
    altered_hash = bytearray(original).copy()

    altered_hash[altered_position] = altered_hash[altered_position] ^ (1 <<
 bit_shift)
    return bytes(altered_hash)


def compare_hash_values(hash1, hash2):
    """
    Compare two hash values by converting them into byte arrays, assuming they
 are little endian.

    :type hash1: bytes
    :param hash1: The hash value to compare.

    :type hash2: bytes
    :param hash2: The hash value to compare.

    :rtype: int
```

```
    :return: Zero if the hash values are equal, otherwise return the difference
 of the first pair of non-matching bytes.
    """
    assert len(hash1) == HASH_LENGTH
    assert len(hash2) == HASH_LENGTH

    hash_array1 = array('b', hash1)
    hash_array2 = array('b', hash2)

    for i in range(len(hash_array1) - 1, -1, -1):
        difference = hash_array1[i] - hash_array2[i]
        if difference != 0:
            return difference
    return 0


def join_hash_pairwise(hash1, hash2):
    """
    Take two hash values, sort them, concatenate them, and generate a new hash
 value from the concatenated values.

    :type hash1: bytes
    :param hash1: Hash value to concatenate.

    :type hash2: bytes
    :param hash2: Hash value to concatenate.

    :rtype: bytes
    :return: The new hash value generated from concatenated hash values.
    """
    if len(hash1) == 0:
        return hash2
    if len(hash2) == 0:
        return hash1

    concatenated = hash1 + hash2 if compare_hash_values(hash1, hash2) < 0 else
 hash2 + hash1
    new_hash_lib = sha256()
    new_hash_lib.update(concatenated)
    new_digest = new_hash_lib.digest()
    return new_digest


def calculate_root_hash_from_internal_hashes(internal_hashes, leaf_hash):
```

```
    """
    Combine the internal hashes and the leaf hash until only one root hash
 remains.

    :type internal_hashes: map
    :param internal_hashes: An iterable over a list of hash values.

    :type leaf_hash: bytes
    :param leaf_hash: The revision hash to pair with the first hash in the Proof
 hashes list.

    :rtype: bytes
    :return: The root hash constructed by combining internal hashes.
    """
    root_hash = reduce(join_hash_pairwise, internal_hashes, leaf_hash)
    return root_hash


def build_candidate_digest(proof, leaf_hash):
    """
    Build the candidate digest representing the entire ledger from the Proof
 hashes.

    :type proof: dict
    :param proof: The Proof object.

    :type leaf_hash: bytes
    :param leaf_hash: The revision hash to pair with the first hash in the Proof
 hashes list.

    :rtype: bytes
    :return: The calculated root hash.
    """
    parsed_proof = parse_proof(proof)
    root_hash = calculate_root_hash_from_internal_hashes(parsed_proof, leaf_hash)
    return root_hash


def verify_document(document_hash, digest, proof):
    """
    Verify document revision against the provided digest.

    :type document_hash: bytes
```

```
    :param document_hash: The SHA-256 value representing the document revision to
 be verified.

    :type digest: bytes
    :param digest: The SHA-256 hash value representing the ledger digest.

    :type proof: dict
    :param proof: The Proof object retrieved
 from :func:`pyqldbsamples.get_revision.get_revision`.

    :rtype: bool
    :return: If the document revision verify against the ledger digest.
    """
    candidate_digest = build_candidate_digest(proof, document_hash)
    return digest == candidate_digest


def to_base_64(input):
    """
    Encode input in base64.

    :type input: bytes
    :param input: Input to be encoded.

    :rtype: string
    :return: Return input that has been encoded in base64.
    """
    encoded_value = b64encode(input)
    return str(encoded_value, 'UTF-8')
```

3. qldb_string_utils.py

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
# permit persons to whom the Software is furnished to do so.
```

```
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
from amazon.ion.simpleion import dumps, loads


def value_holder_to_string(value_holder):
    """
    Returns the string representation of a given `value_holder`.

    :type value_holder: dict
    :param value_holder: The `value_holder` to convert to string.

    :rtype: str
    :return: The string representation of the supplied `value_holder`.
    """
    ret_val = dumps(loads(value_holder), binary=False, indent='  ',
 omit_version_marker=True)
    val = '{{ IonText: {}}}'.format(ret_val)
    return val


def block_response_to_string(block_response):
    """
    Returns the string representation of a given `block_response`.

    :type block_response: dict
    :param block_response: The `block_response` to convert to string.

    :rtype: str
    :return: The string representation of the supplied `block_response`.
    """
    string = ''
    if block_response.get('Block', {}).get('IonText') is not None:
        string += 'Block: ' + value_holder_to_string(block_response['Block']
['IonText']) + ', '
```

```
    if block_response.get('Proof', {}).get('IonText') is not None:
        string += 'Proof: ' + value_holder_to_string(block_response['Proof']
['IonText'])

    return '{' + string + '}'


def digest_response_to_string(digest_response):
    """
    Returns the string representation of a given `digest_response`.

    :type digest_response: dict
    :param digest_response: The `digest_response` to convert to string.

    :rtype: str
    :return: The string representation of the supplied `digest_response`.
    """
    string = ''
    if digest_response.get('Digest') is not None:
        string += 'Digest: ' + str(digest_response['Digest']) + ', '

    if digest_response.get('DigestTipAddress', {}).get('IonText') is not None:
        string += 'DigestTipAddress: ' +
 value_holder_to_string(digest_response['DigestTipAddress']['IonText'])

    return '{' + string + '}'
```

2. Utilisez deux `.py` programmes (`get_digest.py`et`get_revision.py`) pour effectuer les opérations suivantes :

   - Demandez un nouveau résumé depuis le`vehicle-registration` registre.
   - Demandez une preuve pour chaque révision du document avec le VIN`1N4AL11D75C109151` à partir du`VehicleRegistration` tableau.
   - Vérifiez les révisions à l'aide du condensé renvoyé et corrigez-le en recalculant le résumé.

   Le`get_digest.py` programme contient le code suivant.

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
```

```
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from boto3 import client

from pyqldbsamples.constants import Constants
from pyqldbsamples.qldb.qldb_string_utils import digest_response_to_string

logger = getLogger(__name__)
basicConfig(level=INFO)
qldb_client = client('qldb')


def get_digest_result(name):
    """
    Get the digest of a ledger's journal.

    :type name: str
    :param name: Name of the ledger to operate on.

    :rtype: dict
    :return: The digest in a 256-bit hash value and a block address.
    """
    logger.info("Let's get the current digest of the ledger named {}".format(name))
    result = qldb_client.get_digest(Name=name)
```

```
    logger.info('Success. LedgerDigest:
 {}.'.format(digest_response_to_string(result)))
    return result


def main(ledger_name=Constants.LEDGER_NAME):
    """
    This is an example for retrieving the digest of a particular ledger.
    """
    try:
        get_digest_result(ledger_name)
    except Exception as e:
        logger.exception('Unable to get a ledger digest!')
        raise e


if __name__ == '__main__':
    main()
```

> **Note**
>
> Utilisez cetteget_digest_result fonction pour demander un résumé qui couvre la partie actuelle du journal dans votre grand livre. La pointe du journal fait référence au dernier bloc validé au moment où QLDB reçoit votre demande.

Legetrevision.py programme contient le code suivant.

3.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
```

```
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from amazon.ion.simpleion import loads
from boto3 import client

from pyqldbsamples.constants import Constants
from pyqldbsamples.get_digest import get_digest_result
from pyqldbsamples.model.sample_data import SampleData, convert_object_to_ion
from pyqldbsamples.qldb.block_address import block_address_to_dictionary
from pyqldbsamples.verifier import verify_document, flip_random_bit, to_base_64
from pyqldbsamples.connect_to_ledger import create_qldb_driver
from pyqldbsamples.qldb.qldb_string_utils import value_holder_to_string

logger = getLogger(__name__)
basicConfig(level=INFO)
qldb_client = client('qldb')


def get_revision(ledger_name, document_id, block_address, digest_tip_address):
    """
    Get the revision data object for a specified document ID and block address.
    Also returns a proof of the specified revision for verification.

    :type ledger_name: str
    :param ledger_name: Name of the ledger containing the document to query.

    :type document_id: str
    :param document_id: Unique ID for the document to be verified, contained in
 the committed view of the document.

    :type block_address: dict
```

```
    :param block_address: The location of the block to request.

    :type digest_tip_address: dict
    :param digest_tip_address: The latest block location covered by the digest.

    :rtype: dict
    :return: The response of the request.
    """
    result = qldb_client.get_revision(Name=ledger_name,
 BlockAddress=block_address, DocumentId=document_id,
                                      DigestTipAddress=digest_tip_address)
    return result


def lookup_registration_for_vin(driver, vin):
    """
    Query revision history for a particular vehicle for verification.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type vin: str
    :param vin: VIN to query the revision history of a specific registration
 with.

    :rtype: :py:class:`pyqldb.cursor.buffered_cursor.BufferedCursor`
    :return: Cursor on the result set of the statement query.
    """
    logger.info("Querying the 'VehicleRegistration' table for VIN:
 {}...".format(vin))
    query = 'SELECT * FROM _ql_committed_VehicleRegistration WHERE data.VIN = ?'
    return driver.execute_lambda(lambda txn: txn.execute_statement(query,
 convert_object_to_ion(vin)))


def verify_registration(driver, ledger_name, vin):
    """
    Verify each version of the registration for the given VIN.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type ledger_name: str
    :param ledger_name: The ledger to get digest from.
```

```
    :type vin: str
    :param vin: VIN to query the revision history of a specific registration
with.

    :raises AssertionError: When verification failed.
    """
    logger.info("Let's verify the registration with VIN = {}, in ledger =
{}.".format(vin, ledger_name))
    digest = get_digest_result(ledger_name)
    digest_bytes = digest.get('Digest')
    digest_tip_address = digest.get('DigestTipAddress')

    logger.info('Got a ledger digest: digest tip address = {}, digest =
{}.'.format(
        value_holder_to_string(digest_tip_address.get('IonText')),
to_base_64(digest_bytes)))

    logger.info('Querying the registration with VIN = {} to verify each version
of the registration...'.format(vin))
    cursor = lookup_registration_for_vin(driver, vin)
    logger.info('Getting a proof for the document.')

    for row in cursor:
        block_address = row.get('blockAddress')
        document_id = row.get('metadata').get('id')

        result = get_revision(ledger_name, document_id,
block_address_to_dictionary(block_address), digest_tip_address)
        revision = result.get('Revision').get('IonText')
        document_hash = loads(revision).get('hash')

        proof = result.get('Proof')
        logger.info('Got back a proof: {}.'.format(proof))

        verified = verify_document(document_hash, digest_bytes, proof)
        if not verified:
            raise AssertionError('Document revision is not verified.')
        else:
            logger.info('Success! The document is verified.')

        altered_document_hash = flip_random_bit(document_hash)
        logger.info("Flipping one bit in the document's hash and assert that the
document is NOT verified. "
```

```
                    "The altered document hash is:
 {}.".format(to_base_64(altered_document_hash)))
        verified = verify_document(altered_document_hash, digest_bytes, proof)
        if verified:
            raise AssertionError('Expected altered document hash to not be
 verified against digest.')
        else:
            logger.info('Success! As expected flipping a bit in the document
 hash causes verification to fail.')

        logger.info('Finished verifying the registration with VIN = {} in ledger
 = {}.'.format(vin, ledger_name))


def main(ledger_name=Constants.LEDGER_NAME):
    """
    Verify the integrity of a document revision in a QLDB ledger.
    """
    registration = SampleData.VEHICLE_REGISTRATION[0]
    vin = registration['VIN']
    try:
        with create_qldb_driver(ledger_name) as driver:
            verify_registration(driver, ledger_name, vin)
    except Exception as e:
        logger.exception('Unable to verify revision.')
        raise e


if __name__ == '__main__':
    main()
```

2.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
```

```
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from amazon.ion.simpleion import loads
from boto3 import client

from pyqldbsamples.constants import Constants
from pyqldbsamples.get_digest import get_digest_result
from pyqldbsamples.model.sample_data import SampleData, convert_object_to_ion
from pyqldbsamples.qldb.block_address import block_address_to_dictionary
from pyqldbsamples.verifier import verify_document, flip_random_bit, to_base_64
from pyqldbsamples.connect_to_ledger import create_qldb_session
from pyqldbsamples.qldb.qldb_string_utils import value_holder_to_string

logger = getLogger(__name__)
basicConfig(level=INFO)
qldb_client = client('qldb')


def get_revision(ledger_name, document_id, block_address, digest_tip_address):
    """
    Get the revision data object for a specified document ID and block address.
    Also returns a proof of the specified revision for verification.

    :type ledger_name: str
    :param ledger_name: Name of the ledger containing the document to query.

    :type document_id: str
```

```
    :param document_id: Unique ID for the document to be verified, contained in
 the committed view of the document.

    :type block_address: dict
    :param block_address: The location of the block to request.

    :type digest_tip_address: dict
    :param digest_tip_address: The latest block location covered by the digest.

    :rtype: dict
    :return: The response of the request.
    """
    result = qldb_client.get_revision(Name=ledger_name,
 BlockAddress=block_address, DocumentId=document_id,
                                      DigestTipAddress=digest_tip_address)
    return result


def lookup_registration_for_vin(qldb_session, vin):
    """
    Query revision history for a particular vehicle for verification.

    :type qldb_session: :py:class:`pyqldb.session.qldb_session.QldbSession`
    :param qldb_session: An instance of the QldbSession class.

    :type vin: str
    :param vin: VIN to query the revision history of a specific registration
 with.

    :rtype: :py:class:`pyqldb.cursor.buffered_cursor.BufferedCursor`
    :return: Cursor on the result set of the statement query.
    """
    logger.info("Querying the 'VehicleRegistration' table for VIN:
 {}...".format(vin))
    query = 'SELECT * FROM _ql_committed_VehicleRegistration WHERE data.VIN = ?'
    parameters = [convert_object_to_ion(vin)]
    cursor = qldb_session.execute_statement(query, parameters)
    return cursor


def verify_registration(qldb_session, ledger_name, vin):
    """
    Verify each version of the registration for the given VIN.
```

```
    :type qldb_session: :py:class:`pyqldb.session.qldb_session.QldbSession`
    :param qldb_session: An instance of the QldbSession class.

    :type ledger_name: str
    :param ledger_name: The ledger to get digest from.

    :type vin: str
    :param vin: VIN to query the revision history of a specific registration
with.

    :raises AssertionError: When verification failed.
    """
    logger.info("Let's verify the registration with VIN = {}, in ledger =
{}.".format(vin, ledger_name))
    digest = get_digest_result(ledger_name)
    digest_bytes = digest.get('Digest')
    digest_tip_address = digest.get('DigestTipAddress')

    logger.info('Got a ledger digest: digest tip address = {}, digest =
{}.'.format(
        value_holder_to_string(digest_tip_address.get('IonText')),
to_base_64(digest_bytes)))

    logger.info('Querying the registration with VIN = {} to verify each version
of the registration...'.format(vin))
    cursor = lookup_registration_for_vin(qldb_session, vin)
    logger.info('Getting a proof for the document.')

    for row in cursor:
        block_address = row.get('blockAddress')
        document_id = row.get('metadata').get('id')

        result = get_revision(ledger_name, document_id,
block_address_to_dictionary(block_address), digest_tip_address)
        revision = result.get('Revision').get('IonText')
        document_hash = loads(revision).get('hash')

        proof = result.get('Proof')
        logger.info('Got back a proof: {}.'.format(proof))

        verified = verify_document(document_hash, digest_bytes, proof)
        if not verified:
            raise AssertionError('Document revision is not verified.')
        else:
```

```
            logger.info('Success! The document is verified.')

        altered_document_hash = flip_random_bit(document_hash)
        logger.info("Flipping one bit in the document's hash and assert that the
 document is NOT verified. "
                    "The altered document hash is:
 {}.".format(to_base_64(altered_document_hash)))
        verified = verify_document(altered_document_hash, digest_bytes, proof)
        if verified:
            raise AssertionError('Expected altered document hash to not be
 verified against digest.')
        else:
            logger.info('Success! As expected flipping a bit in the document
 hash causes verification to fail.')

        logger.info('Finished verifying the registration with VIN = {} in ledger
 = {}.'.format(vin, ledger_name))


if __name__ == '__main__':
    """
    Verify the integrity of a document revision in a QLDB ledger.
    """
    registration = SampleData.VEHICLE_REGISTRATION[0]
    vin = registration['VIN']
    try:
        with create_qldb_session() as session:
            verify_registration(session, Constants.LEDGER_NAME, vin)
    except Exception:
        logger.exception('Unable to verify revision.')
```

> **Note**
>
> Une fois que la`get_revision` fonction a renvoyé une preuve pour la révision du document spécifiée, ce programme utilise une API côté client pour vérifier cette révision.

3. Pour exécuter le programme, saisissez la commande suivante.

```
python get_revision.py
```

Si vous n'avez plus besoin d'utiliser le`vehicle-registration` registre, passez àÉtape 8 (optionnelle) : nettoyer les ressources.

Étape 8 (optionnelle) : nettoyer les ressources

Vous pouvez continuer à utiliser le`vehicle-registration` registre. Toutefois, si vous n'en avez plus besoin, vous devez le supprimer.

Pour supprimer le registre

1. Utilisez le programme suivant (`delete_ledger.py`) pour supprimer votre`vehicle-registration` registre et tout son contenu.

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO
from time import sleep

from boto3 import client

from pyqldbsamples.constants import Constants
```

```
from pyqldbsamples.describe_ledger import describe_ledger

logger = getLogger(__name__)
basicConfig(level=INFO)
qldb_client = client('qldb')

LEDGER_DELETION_POLL_PERIOD_SEC = 20


def delete_ledger(ledger_name):
    """
    Send a request to QLDB to delete the specified ledger.

    :type ledger_name: str
    :param ledger_name: Name for the ledger to be deleted.

    :rtype: dict
    :return: Result from the request.
    """
    logger.info('Attempting to delete the ledger with name:
 {}...'.format(ledger_name))
    result = qldb_client.delete_ledger(Name=ledger_name)
    logger.info('Success.')
    return result


def wait_for_deleted(ledger_name):
    """
    Wait for the ledger to be deleted.

    :type ledger_name: str
    :param ledger_name: The ledger to check on.
    """
    logger.info('Waiting for the ledger to be deleted...')
    while True:
        try:
            describe_ledger(ledger_name)
            logger.info('The ledger is still being deleted. Please wait...')
            sleep(LEDGER_DELETION_POLL_PERIOD_SEC)
        except qldb_client.exceptions.ResourceNotFoundException:
            logger.info('Success. The ledger is deleted.')
            break
```

```
def set_deletion_protection(ledger_name, deletion_protection):
    """
    Update an existing ledger's deletion protection.

    :type ledger_name: str
    :param ledger_name: Name of the ledger to update.

    :type deletion_protection: bool
    :param deletion_protection: Enable or disable the deletion protection.

    :rtype: dict
    :return: Result from the request.
    """
    logger.info("Let's set deletion protection to {} for the ledger with name
 {}.".format(deletion_protection,

       ledger_name))
    result = qldb_client.update_ledger(Name=ledger_name,
 DeletionProtection=deletion_protection)
    logger.info('Success. Ledger updated: {}'.format(result))


def main(ledger_name=Constants.LEDGER_NAME):
    """
    Delete a ledger.
    """
    try:
        set_deletion_protection(ledger_name, False)
        delete_ledger(ledger_name)
        wait_for_deleted(ledger_name)
    except Exception as e:
        logger.exception('Unable to delete the ledger.')
        raise e


if __name__ == '__main__':
    main()
```

> **Note**
>
> Si la protection contre la suppression est activée pour votre registre, vous devez commencer par la désactiver avant de pouvoir supprimer le registre à l'aide de l'API QLDB.

Le `delete_ledger.py` fichier dépend également du programme suivant (`describe_ledger.py`).

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from boto3 import client

from pyqldbsamples.constants import Constants

logger = getLogger(__name__)
```

```
basicConfig(level=INFO)
qldb_client = client('qldb')


def describe_ledger(ledger_name):
    """
    Describe a ledger.

    :type ledger_name: str
    :param ledger_name: Name of the ledger to describe.
    """
    logger.info('describe ledger with name: {}.'.format(ledger_name))
    result = qldb_client.describe_ledger(Name=ledger_name)
    result.pop('ResponseMetadata')
    logger.info('Success. Ledger description: {}.'.format(result))
    return result


def main(ledger_name=Constants.LEDGER_NAME):
    """
    Describe a QLDB ledger.
    """
    try:
        describe_ledger(ledger_name)
    except Exception as e:
        logger.exception('Unable to describe a ledger.')
        raise e


if __name__ == '__main__':
    main()
```

2. Pour exécuter le programme, saisissez la commande suivante.

```
python delete_ledger.py
```

Utilisation des types de données Amazon Ion dans Amazon QLDB

Amazon QLDB stocke ses données au format Amazon Ion. Pour travailler avec des données dans QLDB, vous devez utiliser une bibliothèque Ion comme dépendance pour un langage de programmation pris en charge.

Dans cette section, découvrez comment convertir les données des types natifs en leurs équivalents ioniques, et inversement. Ce guide de référence présente des exemples de code qui utilisent le pilote QLDB pour traiter les données Ion dans un registre QLDB. Il inclut des exemples de code pour Java, .NET (C#), Go, Node.js (C#TypeScript).

Rubriques

Prérequis

Les exemples de code suivants supposent que vous avez une instance de pilote QLDB connectée à un registre actif avec une table nomméeExampleTable. Le tableau contient un seul document existant comportant les huit champs suivants :

- ExampleBool
- ExampleInt
- ExampleFloat
- ExampleDecimal
- ExampleTimestamp
- ExampleString
- ExampleBlob
- ExampleList

> **Note**
>
> Aux fins de cette référence, supposons que le type stocké dans chaque champ correspond à son nom. En pratique, QLDB n'applique pas les définitions de schéma ou de type de données pour les champs du document.

Booléen

Les exemples de code suivants montrent comment traiter le type booléen Ion.

Java

```
// Instantiate an IonSystem from the Ion library
IonSystem ionSystem = IonSystemBuilder.standard().build();

boolean exampleBoolean = driver.execute((txn) -> {
    // Transforming a Java boolean to Ion
    boolean aBoolean = true;
    IonValue ionBool = ionSystem.newBool(aBoolean);

    // Insertion into QLDB
    txn.execute("UPDATE ExampleTable SET ExampleBool = ?", ionBool);

    // Fetching from QLDB
    Result result = txn.execute("SELECT VALUE ExampleBool from ExampleTable");
    // Assume there is only one document in ExampleTable
    for (IonValue ionValue : result)
    {
        // Transforming Ion to a Java boolean
        // Cast IonValue to IonBool first
        aBoolean = ((IonBool)ionValue).booleanValue();
    }

    // exampleBoolean is now the value fetched from QLDB
    return aBoolean;
});
```

.NET

```
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;
...
```

```
IValueFactory valueFactory = new ValueFactory();

// Transforming a C# bool to Ion.
bool nativeBool  = true;
IIonValue ionBool = valueFactory.NewBool(nativeBool);

IAsyncResult selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
    await txn.Execute("UPDATE ExampleTable SET ExampleBool = ?", ionBool);

    // Fetching from QLDB.
    return await txn.Execute("SELECT VALUE ExampleBool from ExampleTable");
});

bool? retrievedBool = null;

// Assume there is only one document in ExampleTable.
await foreach (IIonValue ionValue in selectResult)
{
    // Transforming Ion to a C# bool.
    retrievedBool = ionValue.BoolValue;
}
```

> **Note**
>
> Pour convertir en code synchrone, supprimez les`async` mots-clés`await` et et remplacez le`IAsyncResult` type par`IResult`.

Go

```
exampleBool, err := driver.Execute(context.Background(), func(txn
 qldbdriver.Transaction) (interface{}, error) {
 aBool := true

 // Insertion into QLDB
 _, err = txn.Execute("UPDATE ExampleTable SET ExampleBool = ?", aBool)
 if err != nil {
  return nil, err
 }
```

```
 // Fetching from QLDB
 result, err := txn.Execute("SELECT VALUE ExampleBool FROM ExampleTable")
 if err != nil {
  return nil, err
 }

 // Assume there is only one document in ExampleTable
 if result.Next(txn) {
  var decodedResult bool
  err := ion.Unmarshal(result.GetCurrentData(), &decodedResult)
  if err != nil {
   return nil, err
  }

  // exampleBool is now the value fetched from QLDB
  return decodedResult, nil
 }
 return nil, result.Err()
})
```

Node.js

```
async function queryIonBoolean(driver: QldbDriver): Promise<boolean> {
    return (driver.executeLambda(async (txn: TransactionExecutor) => {
            // Updating QLDB
            await txn.execute("UPDATE ExampleTable SET ExampleBool = ?", true);

            // Fetching from QLDB
            const resultList: dom.Value[] = (await txn.execute("SELECT VALUE
 ExampleBool FROM ExampleTable")).getResultList();

            // Assume there is only one document in ExampleTable
            const ionValue: dom.Value = resultList[0];
            // Transforming Ion to a TypeScript Boolean
            const boolValue: boolean = ionValue.booleanValue();
            return boolValue;
        })
    );
}
```

Python

```
def update_and_query_ion_bool(txn):
    # QLDB can take in a Python bool
    a_bool = True

    # Insertion into QLDB
    txn.execute_statement("UPDATE ExampleTable SET ExampleBool = ?", a_bool)

    # Fetching from QLDB
    cursor = txn.execute_statement("SELECT VALUE ExampleBool FROM ExampleTable")

    # Assume there is only one document in ExampleTable
    for ion_value in cursor:
        # Ion Python bool is a child class of Python bool
        a_bool = ion_value

    # example_bool is now the value fetched from QLDB
    return a_bool


example_bool = driver.execute_lambda(lambda txn: update_and_query_ion_bool(txn))
```

Int

Les exemples de code suivants montrent comment traiter le type entier Ion.

Java

```
// Instantiate an IonSystem from the Ion library
IonSystem ionSystem = IonSystemBuilder.standard().build();

int exampleInt = driver.execute((txn) -> {
    // Transforming a Java int to Ion
    int aInt = 256;
    IonValue ionInt = ionSystem.newInt(aInt);

    // Insertion into QLDB
    txn.execute("UPDATE ExampleTable SET ExampleInt = ?", ionInt);

    // Fetching from QLDB
    Result result = txn.execute("SELECT VALUE ExampleInt from ExampleTable");
```

```
    // Assume there is only one document in ExampleTable
    for (IonValue ionValue : result)
    {
        // Transforming Ion to a Java int
        // Cast IonValue to IonInt first
        aInt = ((IonInt)ionValue).intValue();
    }

    // exampleInt is now the value fetched from QLDB
    return aInt;
});
```

.NET

```
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;
...

IValueFactory valueFactory = new ValueFactory();

// Transforming a C# int to Ion.
int nativeInt  = 256;
IIonValue ionInt = valueFactory.NewInt(nativeInt);

IAsyncResult selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
    await txn.Execute("UPDATE ExampleTable SET ExampleInt = ?", ionInt);

    // Fetching from QLDB.
    return await txn.Execute("SELECT VALUE ExampleInt from ExampleTable");
});

int? retrievedInt = null;

// Assume there is only one document in ExampleTable.
await foreach (IIonValue ionValue in selectResult)
{
    // Transforming Ion to a C# int.
    retrievedInt = ionValue.IntValue;
}
```

> **Note**
>
> Pour convertir en code synchrone, supprimez les`async` mots-clés`await` et et remplacez le`IAsyncResult` type par`IResult`.

Go

```
exampleInt, err := driver.Execute(context.Background(), func(txn
 qldbdriver.Transaction) (interface{}, error) {
 aInt := 256

 // Insertion into QLDB
 _, err = txn.Execute("UPDATE ExampleTable SET ExampleInt = ?", aInt)
 if err != nil {
  return nil, err
 }

 // Fetching from QLDB
 result, err := txn.Execute("SELECT VALUE ExampleInt FROM ExampleTable")
 if err != nil {
  return nil, err
 }

 // Assume there is only one document in ExampleTable
 if result.Next(txn) {
  var decodedResult int
  err := ion.Unmarshal(result.GetCurrentData(), &decodedResult)
  if err != nil {
   return nil, err
  }

  // exampleInt is now the value fetched from QLDB
  return decodedResult, nil
 }
 return nil, result.Err()
})
```

Node.js

```
async function queryIonInt(driver: QldbDriver): Promise<number> {
    return (driver.executeLambda(async (txn: TransactionExecutor) => {
```

```
            // Updating QLDB
            await txn.execute("UPDATE ExampleTable SET ExampleInt = ?", 256);

            // Fetching from QLDB
            const resultList: dom.Value[] = (await txn.execute("SELECT VALUE
 ExampleInt FROM ExampleTable")).getResultList();

            // Assume there is only one document in ExampleTable
            const ionValue: dom.Value = resultList[0];
            // Transforming Ion to a TypeScript Number
            const intValue: number = ionValue.numberValue();
            return intValue;
        })
    );
}
```

Python

```
def update_and_query_ion_int(txn):
    # QLDB can take in a Python int
    a_int = 256

    # Insertion into QLDB
    txn.execute_statement("UPDATE ExampleTable SET ExampleInt = ?", a_int)

    # Fetching from QLDB
    cursor = txn.execute_statement("SELECT VALUE ExampleInt FROM ExampleTable")

    # Assume there is only one document in ExampleTable
    for ion_value in cursor:
        # Ion Python int is a child class of Python int
        a_int = ion_value

    # example_int is now the value fetched from QLDB
    return a_int


example_int = driver.execute_lambda(lambda txn: update_and_query_ion_int(txn))
```

Float

Les exemples de code suivants montrent comment traiter le type Ion float.

Java

```
// Instantiate an IonSystem from the Ion library
IonSystem ionSystem = IonSystemBuilder.standard().build();

float exampleFloat = driver.execute((txn) -> {
    // Transforming a Java float to Ion
    float aFloat = 256;
    IonValue ionFloat = ionSystem.newFloat(aFloat);

    // Insertion into QLDB
    txn.execute("UPDATE ExampleTable SET ExampleFloat = ?", ionFloat);

    // Fetching from QLDB
    Result result = txn.execute("SELECT VALUE ExampleFloat from ExampleTable");
    // Assume there is only one document in ExampleTable
    for (IonValue ionValue : result)
    {
        // Transforming Ion to a Java float
        // Cast IonValue to IonFloat first
        aFloat = ((IonFloat)ionValue).floatValue();
    }

    // exampleFloat is now the value fetched from QLDB
    return aFloat;
});
```

.NET

Utiliser C# float

```
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;
...

IValueFactory valueFactory = new ValueFactory();

// Transforming a C# float to Ion.
float nativeFloat = 256;
IIonValue ionFloat = valueFactory.NewFloat(nativeFloat);

IAsyncResult selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
```

```
    await txn.Execute("UPDATE ExampleTable SET ExampleFloat = ?", ionFloat);

    // Fetching from QLDB.
    return await txn.Execute("SELECT VALUE ExampleFloat from ExampleTable");
});

float? retrievedFloat = null;

// Assume there is only one document in ExampleTable.
await foreach (IIonValue ionValue in selectResult)
{
    // Transforming Ion to a C# float. We cast ionValue.DoubleValue to a float
    // but be cautious, this is a down-cast and can lose precision.
    retrievedFloat = (float)ionValue.DoubleValue;
}
```

> **Note**
>
> Pour convertir en code synchrone, supprimez les`async` mots-clés`await` et et remplacez le`IAsyncResult` type par`IResult`.

Utiliser C# double

```
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;
...

IValueFactory valueFactory = new ValueFactory();

// Transforming a C# double to Ion.
double nativeDouble = 256;
IIonValue ionFloat = valueFactory.NewFloat(nativeDouble);

IAsyncResult selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
    await txn.Execute("UPDATE ExampleTable SET ExampleFloat = ?", ionFloat);

    // Fetching from QLDB.
    return await txn.Execute("SELECT VALUE ExampleFloat from ExampleTable");
});
```

```
double? retrievedDouble = null;

// Assume there is only one document in ExampleTable.
await foreach (IIonValue ionValue in selectResult)
{
    // Transforming Ion to a C# double.
    retrievedDouble = ionValue.DoubleValue;
}
```

> **Note**
>
> Pour convertir en code synchrone, supprimez les`async` mots-clés`await` et et remplacez le`IAsyncResult` type par`IResult`.

Go

```
exampleFloat, err := driver.Execute(context.Background(), func(txn
 qldbdriver.Transaction) (interface{}, error) {
 aFloat := float32(256)

 // Insertion into QLDB
 _, err = txn.Execute("UPDATE ExampleTable SET ExampleFloat = ?", aFloat)
 if err != nil {
  return nil, err
 }

 // Fetching from QLDB
 result, err := txn.Execute("SELECT VALUE ExampleFloat FROM ExampleTable")
 if err != nil {
  return nil, err
 }

 // Assume there is only one document in ExampleTable
 if result.Next(txn) {
  // float64 would work as well
  var decodedResult float32
  err := ion.Unmarshal(result.GetCurrentData(), &decodedResult)
  if err != nil {
   return nil, err
  }
```

```
  // exampleFloat is now the value fetched from QLDB
  return decodedResult, nil
 }
 return nil, result.Err()
})
```

Node.js

```
async function queryIonFloat(driver: QldbDriver): Promise<number> {
    return (driver.executeLambda(async (txn: TransactionExecutor) => {
            // Updating QLDB
            await txn.execute("UPDATE ExampleTable SET ExampleFloat = ?", 25.6);

            // Fetching from QLDB
            const resultList: dom.Value[] = (await txn.execute("SELECT VALUE
 ExampleFloat FROM ExampleTable")).getResultList();

            // Assume there is only one document in ExampleTable
            const ionValue: dom.Value = resultList[0];
            // Transforming Ion to a TypeScript Number
            const floatValue: number = ionValue.numberValue();
            return floatValue;
        })
    );
}
```

Python

```
def update_and_query_ion_float(txn):
    # QLDB can take in a Python float
    a_float = float(256)

    # Insertion into QLDB
    txn.execute_statement("UPDATE ExampleTable SET ExampleFloat = ?", a_float)

    # Fetching from QLDB
    cursor = txn.execute_statement("SELECT VALUE ExampleFloat FROM ExampleTable")

    # Assume there is only one document in ExampleTable
    for ion_value in cursor:
        # Ion Python float is a child class of Python float
        a_float = ion_value
```

```
    # example_float is now the value fetched from QLDB
    return a_float


example_float = driver.execute_lambda(lambda txn: update_and_query_ion_float(txn))
```

Décimal

Les exemples de code suivants montrent comment traiter le type décimal Ion.

Java

```
// Instantiate an IonSystem from the Ion library
IonSystem ionSystem = IonSystemBuilder.standard().build();

double exampleDouble = driver.execute((txn) -> {
    // Transforming a Java double to Ion
    double aDouble = 256;
    IonValue ionDecimal = ionSystem.newDecimal(aDouble);

    // Insertion into QLDB
    txn.execute("UPDATE ExampleTable SET ExampleDecimal = ?", ionDecimal);

    // Fetching from QLDB
    Result result = txn.execute("SELECT VALUE ExampleDecimal from ExampleTable");
    // Assume there is only one document in ExampleTable
    for (IonValue ionValue : result)
    {
        // Transforming Ion to a Java double
        // Cast IonValue to IonDecimal first
        aDouble = ((IonDecimal)ionValue).doubleValue();
    }

    // exampleDouble is now the value fetched from QLDB
    return aDouble;
});
```

.NET

```
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;
...
```

```
IValueFactory valueFactory = new ValueFactory();

// Transforming a C# decimal to Ion.
decimal nativeDecimal = 256.8723m;
IIonValue ionDecimal = valueFactory.NewDecimal(nativeDecimal);

IAsyncResult selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
    await txn.Execute("UPDATE ExampleTable SET ExampleDecimal = ?", ionDecimal);

    // Fetching from QLDB.
    return await txn.Execute("SELECT VALUE ExampleDecimal from ExampleTable");
});

decimal? retrievedDecimal = null;

// Assume there is only one document in ExampleTable.
await foreach (IIonValue ionValue in selectResult)
{
    // Transforming Ion to a C# decimal.
    retrievedDecimal = ionValue.DecimalValue;
}
```

> **Note**
>
> Pour convertir en code synchrone, supprimez les`async` mots-clés`await` et et remplacez le`IAsyncResult` type par`IResult`.

Go

```
exampleDecimal, err := driver.Execute(context.Background(), func(txn
 qldbdriver.Transaction) (interface{}, error) {
 aDecimal, err := ion.ParseDecimal("256")
 if err != nil {
  return nil, err
 }

 // Insertion into QLDB
 _, err = txn.Execute("UPDATE ExampleTable SET ExampleDecimal = ?", aDecimal)
 if err != nil {
```

```
  return nil, err
 }

 // Fetching from QLDB
 result, err := txn.Execute("SELECT VALUE ExampleDecimal FROM ExampleTable")
 if err != nil {
  return nil, err
 }

 // Assume there is only one document in ExampleTable
 if result.Next(txn) {
  var decodedResult ion.Decimal
  err := ion.Unmarshal(result.GetCurrentData(), &decodedResult)
  if err != nil {
   return nil, err
  }

  // exampleDecimal is now the value fetched from QLDB
  return decodedResult, nil
 }
 return nil, result.Err()
})
```

Node.js

```
async function queryIonDecimal(driver: QldbDriver): Promise<Decimal> {
    return (driver.executeLambda(async (txn: TransactionExecutor) => {
            // Creating a Decimal value. Decimal is an Ion Type with high precision
            let ionDecimal: Decimal = dom.load("2.5d-6").decimalValue();
            // Updating QLDB
            await txn.execute("UPDATE ExampleTable SET ExampleDecimal = ?",
 ionDecimal);

            // Fetching from QLDB
            const resultList: dom.Value[] = (await txn.execute("SELECT VALUE
 ExampleDecimal FROM ExampleTable")).getResultList();

            // Assume there is only one document in ExampleTable
            const ionValue: dom.Value = resultList[0];
            // Get the Ion Decimal
            ionDecimal = ionValue.decimalValue();
            return ionDecimal;
        })
```

```
    );
}
```

> **Note**
>
> Vous pouvez également l'utiliser`ionValue.numberValue()` pour transformer un ion décimal en JavaScript nombre. L'utilisation`ionValue.numberValue()` offre de meilleures performances, mais elle est moins précise.

Python

```
def update_and_query_ion_decimal(txn):
    # QLDB can take in a Python decimal
    a_decimal = Decimal(256)

    # Insertion into QLDB
    txn.execute_statement("UPDATE ExampleTable SET ExampleDecimal = ?", a_decimal)

    # Fetching from QLDB
    cursor = txn.execute_statement("SELECT VALUE ExampleDecimal FROM ExampleTable")

    # Assume there is only one document in ExampleTable
    for ion_value in cursor:
        # Ion Python decimal is a child class of Python decimal
        a_decimal = ion_value

    # example_decimal is now the value fetched from QLDB
    return a_decimal


example_decimal = driver.execute_lambda(lambda txn:
 update_and_query_ion_decimal(txn))
```

Horodatage

Les exemples de code suivants montrent comment traiter le type d'horodatage Ion.

Java

```
// Instantiate an IonSystem from the Ion library
IonSystem ionSystem = IonSystemBuilder.standard().build();

Date exampleDate = driver.execute((txn) -> {
    // Transforming a Java Date to Ion
    Date aDate = new Date();
    IonValue ionTimestamp = ionSystem.newUtcTimestamp(aDate);

    // Insertion into QLDB
    txn.execute("UPDATE ExampleTable SET ExampleTimestamp = ?", ionTimestamp);

    // Fetching from QLDB
    Result result = txn.execute("SELECT VALUE ExampleTimestamp from ExampleTable");
    // Assume there is only one document in ExampleTable
    for (IonValue ionValue : result)
    {
        // Transforming Ion to a Java Date
        // Cast IonValue to IonTimestamp first
        aDate = ((IonTimestamp)ionValue).dateValue();
    }

    // exampleDate is now the value fetched from QLDB
    return aDate;
});
```

.NET

```
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;
using Amazon.IonDotnet;
...

IValueFactory valueFactory = new ValueFactory();

// Convert C# native DateTime to Ion.
DateTime nativeDateTime = DateTime.Now;

// First convert it to a timestamp object from Ion.
Timestamp timestamp = new Timestamp(nativeDateTime);
// Then convert to Ion timestamp.
IIonValue ionTimestamp = valueFactory.NewTimestamp(timestamp);
```

```
IAsyncResult selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
    await txn.Execute("UPDATE ExampleTable SET ExampleTimestamp = ?", ionTimestamp);

    // Fetching from QLDB.
    return await txn.Execute("SELECT VALUE ExampleTimestamp from ExampleTable");
});

DateTime? retrievedDateTime = null;

// Assume there is only one document in ExampleTable.
await foreach (IIonValue ionValue in selectResult)
{
    // Transforming Ion to a C# DateTime.
    retrievedDateTime = ionValue.TimestampValue.DateTimeValue;
}
```

> **Note**
>
> Pour convertir en code synchrone, supprimez les`async` mots-clés`await` et et remplacez le`IAsyncResult` type par`IResult`.

Go

```
exampleTimestamp, err := driver.Execute(context.Background(), func(txn
 qldbdriver.Transaction) (interface{}, error) {
 aTimestamp := time.Date(2006, time.May, 20, 12, 30, 0, 0, time.UTC)
 if err != nil {
  return nil, err
 }

 // Insertion into QLDB
 _, err = txn.Execute("UPDATE ExampleTable SET ExampleTimestamp = ?", aTimestamp)
 if err != nil {
  return nil, err
 }

 // Fetching from QLDB
 result, err := txn.Execute("SELECT VALUE ExampleTimestamp FROM ExampleTable")
 if err != nil {
```

```
  return nil, err
 }

 // Assume there is only one document in ExampleTable
 if result.Next(txn) {
  var decodedResult ion.Timestamp
  err := ion.Unmarshal(result.GetCurrentData(), &decodedResult)
  if err != nil {
   return nil, err
  }

  // exampleTimestamp is now the value fetched from QLDB
  return decodedResult.GetDateTime(), nil
 }
 return nil, result.Err()
})
```

Node.js

```
async function queryIonTimestamp(driver: QldbDriver): Promise<Date> {
    return (driver.executeLambda(async (txn: TransactionExecutor) => {
            let exampleDateTime: Date = new Date(Date.now());
            // Updating QLDB
            await txn.execute("UPDATE ExampleTable SET ExampleTimestamp = ?",
 exampleDateTime);

            // Fetching from QLDB
            const resultList: dom.Value[] = (await txn.execute("SELECT VALUE
 ExampleTimestamp FROM ExampleTable")).getResultList();

            // Assume there is only one document in ExampleTable
            const ionValue: dom.Value = resultList[0];
            // Transforming Ion to a TypeScript Date
            exampleDateTime = ionValue.timestampValue().getDate();
            return exampleDateTime;
        })
    );
}
```

Python

```
def update_and_query_ion_timestamp(txn):
    # QLDB can take in a Python timestamp
```

```
    a_datetime = datetime.now()

    # Insertion into QLDB
    txn.execute_statement("UPDATE ExampleTable SET ExampleTimestamp = ?",
 a_datetime)

    # Fetching from QLDB
    cursor = txn.execute_statement("SELECT VALUE ExampleTimestamp FROM
 ExampleTable")

    # Assume there is only one document in ExampleTable
    for ion_value in cursor:
        # Ion Python timestamp is a child class of Python datetime
        a_timestamp = ion_value

    # example_timestamp is now the value fetched from QLDB
    return a_timestamp


example_timestamp = driver.execute_lambda(lambda txn:
 update_and_query_ion_timestamp(txn))
```

Chaîne

Les exemples de code suivants montrent comment traiter le type de chaîne Ion.

Java

```
// Instantiate an IonSystem from the Ion library
IonSystem ionSystem = IonSystemBuilder.standard().build();

String exampleString = driver.execute((txn) -> {
    // Transforming a Java String to Ion
    String aString = "Hello world!";
    IonValue ionString = ionSystem.newString(aString);

    // Insertion into QLDB
    txn.execute("UPDATE ExampleTable SET ExampleString = ?", ionString);

    // Fetching from QLDB
    Result result = txn.execute("SELECT VALUE ExampleString from ExampleTable");
    // Assume there is only one document in ExampleTable
```

```
    for (IonValue ionValue : result)
    {
        // Transforming Ion to a Java String
        // Cast IonValue to IonString first
        aString = ((IonString)ionValue).stringValue();
    }

    // exampleString is now the value fetched from QLDB
    return aString;
});
```

.NET

```
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;
...

IValueFactory valueFactory = new ValueFactory();

// Convert C# string to Ion.
String nativeString = "Hello world!";
IIonValue ionString = valueFactory.NewString(nativeString);

IAsyncResult selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
    await txn.Execute("UPDATE ExampleTable SET ExampleString = ?", ionString);

    // Fetching from QLDB.
    return await txn.Execute("SELECT VALUE ExampleString from ExampleTable");
});

String retrievedString = null;

// Assume there is only one document in ExampleTable.
await foreach (IIonValue ionValue in selectResult)
{
    // Transforming Ion to a C# string.
    retrievedString = ionValue.StringValue;
}
```

> **Note**
>
> Pour convertir en code synchrone, supprimez lesasync mots-clésawait et et remplacez leIAsyncResult type parIResult.

Go

```
exampleString, err := driver.Execute(context.Background(), func(txn
 qldbdriver.Transaction) (interface{}, error) {
 aString := "Hello World!"

 // Insertion into QLDB
 _, err = txn.Execute("UPDATE ExampleTable SET ExampleString = ?", aString)
 if err != nil {
  return nil, err
 }

 // Fetching from QLDB
 result, err := txn.Execute("SELECT VALUE ExampleString FROM ExampleTable")
 if err != nil {
  return nil, err
 }

 // Assume there is only one document in ExampleTable
 if result.Next(txn) {
  var decodedResult string
  err := ion.Unmarshal(result.GetCurrentData(), &decodedResult)
  if err != nil {
   return nil, err
  }

  // exampleString is now the value fetched from QLDB
  return decodedResult, nil
 }
 return nil, result.Err()
})
```

Node.js

```
async function queryIonString(driver: QldbDriver): Promise<string> {
    return (driver.executeLambda(async (txn: TransactionExecutor) => {
```

```
            // Updating QLDB
            await txn.execute("UPDATE ExampleTable SET ExampleString = ?", "Hello
 World!");

            // Fetching from QLDB
            const resultList: dom.Value[] = (await txn.execute("SELECT VALUE
 ExampleString FROM ExampleTable")).getResultList();

            // Assume there is only one document in ExampleTable
            const ionValue: dom.Value = resultList[0];
            // Transforming Ion to a TypeScript String
            const stringValue: string = ionValue.stringValue();
            return stringValue;
        })
    );
}
```

Python

```
def update_and_query_ion_string(txn):
    # QLDB can take in a Python string
    a_string = "Hello world!"

    # Insertion into QLDB
    txn.execute_statement("UPDATE ExampleTable SET ExampleString = ?", a_string)

    # Fetching from QLDB
    cursor = txn.execute_statement("SELECT VALUE ExampleString FROM ExampleTable")

    # Assume there is only one document in ExampleTable
    for ion_value in cursor:
        # Ion Python string is a child class of Python string
        a_string = ion_value

    # example_string is now the value fetched from QLDB
    return a_string


example_string = driver.execute_lambda(lambda txn: update_and_query_ion_string(txn))
```

BLOB

Les exemples de code suivants montrent comment traiter le type de blob Ion.

Java

```
// Instantiate an IonSystem from the Ion library
IonSystem ionSystem = IonSystemBuilder.standard().build();

byte[] exampleBytes = driver.execute((txn) -> {
    // Transforming a Java byte array to Ion
    // Transform any arbitrary data to a byte array to store in QLDB
    String aString = "Hello world!";
    byte[] aByteArray = aString.getBytes();
    IonValue ionBlob = ionSystem.newBlob(aByteArray);

    // Insertion into QLDB
    txn.execute("UPDATE ExampleTable SET ExampleBlob = ?", ionBlob);

    // Fetching from QLDB
    Result result = txn.execute("SELECT VALUE ExampleBlob from ExampleTable");
    // Assume there is only one document in ExampleTable
    for (IonValue ionValue : result)
    {
        // Transforming Ion to a Java byte array
        // Cast IonValue to IonBlob first
        aByteArray = ((IonBlob)ionValue).getBytes();
    }

    // exampleBytes is now the value fetched from QLDB
    return aByteArray;
});
```

.NET

```
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;
using System.Text;
...

IValueFactory valueFactory = new ValueFactory();

// Transform any arbitrary data to a byte array to store in QLDB.
string nativeString = "Hello world!";
```

```
byte[] nativeByteArray = Encoding.UTF8.GetBytes(nativeString);
// Transforming a C# byte array to Ion.
IIonValue ionBlob = valueFactory.NewBlob(nativeByteArray);

IAsyncResult selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
    await txn.Execute("UPDATE ExampleTable SET ExampleBlob = ?", ionBlob);

    // Fetching from QLDB.
    return await txn.Execute("SELECT VALUE ExampleBlob from ExampleTable");
});

byte[] retrievedByteArray = null;

// Assume there is only one document in ExampleTable.
await foreach (IIonValue ionValue in selectResult)
{
    // Transforming Ion to a C# byte array.
    retrievedByteArray = ionValue.Bytes().ToArray();
}
```

> **Note**
>
> Pour convertir en code synchrone, supprimez les`async` mots-clés`await` et et remplacez le`IAsyncResult` type par`IResult`.

Go

```
exampleBlob, err := driver.Execute(context.Background(), func(txn
 qldbdriver.Transaction) (interface{}, error) {
 aBlob := []byte("Hello World!")

 // Insertion into QLDB
 _, err = txn.Execute("UPDATE ExampleTable SET ExampleBlob = ?", aBlob)
 if err != nil {
  return nil, err
 }

 // Fetching from QLDB
 result, err := txn.Execute("SELECT VALUE ExampleBlob FROM ExampleTable")
```

```
 if err != nil {
  return nil, err
 }

 // Assume there is only one document in ExampleTable
 if result.Next(txn) {
  var decodedResult []byte
  err := ion.Unmarshal(result.GetCurrentData(), &decodedResult)
  if err != nil {
   return nil, err
  }

  // exampleBlob is now the value fetched from QLDB
  return decodedResult, nil
 }
 return nil, result.Err()
})
```

Node.js

```
async function queryIonBlob(driver: QldbDriver): Promise<Uint8Array> {
    return (driver.executeLambda(async (txn: TransactionExecutor) => {
            const enc = new TextEncoder();
            let blobValue: Uint8Array = enc.encode("Hello World!");
            // Updating QLDB
            await txn.execute("UPDATE ExampleTable SET ExampleBlob = ?", blobValue);

            // Fetching from QLDB
            const resultList: dom.Value[] = (await txn.execute("SELECT VALUE
 ExampleBlob FROM ExampleTable")).getResultList();

            // Assume there is only one document in ExampleTable
            const ionValue: dom.Value = resultList[0];
            // Transforming Ion to TypeScript Uint8Array
            blobValue = ionValue.uInt8ArrayValue();
            return blobValue;
        })
    );
}
```

Python

```
def update_and_query_ion_blob(txn):
```

```
    # QLDB can take in a Python byte array
    a_string = "Hello world!"
    a_byte_array = str.encode(a_string)

    # Insertion into QLDB
    txn.execute_statement("UPDATE ExampleTable SET ExampleBlob = ?", a_byte_array)

    # Fetching from QLDB
    cursor = txn.execute_statement("SELECT VALUE ExampleBlob FROM ExampleTable")

    # Assume there is only one document in ExampleTable
    for ion_value in cursor:
        # Ion Python blob is a child class of Python byte array
        a_blob = ion_value

    # example_blob is now the value fetched from QLDB
    return a_blob


example_blob = driver.execute_lambda(lambda txn: update_and_query_ion_blob(txn))
```

Liste

Les exemples de code suivants montrent comment traiter le type de liste d'ions.

Java

```
// Instantiate an IonSystem from the Ion library
IonSystem ionSystem = IonSystemBuilder.standard().build();

List<Integer> exampleList = driver.execute((txn) -> {
    // Transforming a Java List to Ion
    List<Integer> aList = new ArrayList<>();
    // Add 5 Integers to the List for the sake of example
    for (int i = 0; i < 5; i++) {
        aList.add(i);
    }
    // Create an empty Ion List
    IonList ionList = ionSystem.newEmptyList();
    // Add the 5 Integers to the Ion List
    for (Integer i : aList) {
        // Convert each Integer to Ion ints first to add it to the Ion List
```

```
        ionList.add(ionSystem.newInt(i));
    }

    // Insertion into QLDB
    txn.execute("UPDATE ExampleTable SET ExampleList = ?", (IonValue) ionList);

    // Fetching from QLDB
    Result result = txn.execute("SELECT VALUE ExampleList from ExampleTable");
    // Assume there is only one document in ExampleTable
    for (IonValue ionValue : result)
    {
        // Iterate through the Ion List to map it to a Java List
        List<Integer> intList = new ArrayList<>();
        for (IonValue ionInt : (IonList)ionValue) {
            // Convert the 5 Ion ints to Java Integers
            intList.add(((IonInt)ionInt).intValue());
        }
        aList = intList;
    }

    // exampleList is now the value fetched from QLDB
    return aList;
});
```

.NET

```
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;
using System.Collections.Generic;
...

IValueFactory valueFactory = new ValueFactory();

// Transforming a C# list to Ion.
IIonValue ionList = valueFactory.NewEmptyList();
foreach (int i in new List<int> {0, 1, 2, 3, 4})
{
    // Convert to Ion int and add to Ion list.
    ionList.Add(valueFactory.NewInt(i));
}

IAsyncResult selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
```

```
    await txn.Execute("UPDATE ExampleTable SET ExampleList = ?", ionList);

    // Fetching from QLDB.
    return await txn.Execute("SELECT VALUE ExampleList from ExampleTable");
});


List<int> retrievedList = new List<int>();
await foreach (IIonValue ionValue in selectResult)
{
    // Iterate through the Ion List to map it to a C# list.
    foreach (IIonValue ionInt in ionValue)
    {
        retrievedList.Add(ionInt.IntValue);
    }
}
```

> **Note**
>
> Pour convertir en code synchrone, supprimez les`async` mots-clés`await` et et remplacez le`IAsyncResult` type par`IResult`.

Go

```
exampleList, err := driver.Execute(context.Background(), func(txn
 qldbdriver.Transaction) (interface{}, error) {
 aList := []int{1, 2, 3, 4, 5}

 // Insertion into QLDB
 _, err = txn.Execute("UPDATE ExampleTable SET ExampleList = ?", aList)
 if err != nil {
  return nil, err
 }

 // Fetching from QLDB
 result, err := txn.Execute("SELECT VALUE ExampleList FROM ExampleTable")
 if err != nil {
  return nil, err
 }

 // Assume there is only one document in ExampleTable
```

```
 if result.Next(txn) {
  var decodedResult []int
  err := ion.Unmarshal(result.GetCurrentData(), &decodedResult)
  if err != nil {
   return nil, err
  }

  // exampleList is now the value fetched from QLDB
  return decodedResult, nil
 }
 return nil, result.Err()
})
```

Node.js

```
async function queryIonList(driver: QldbDriver): Promise<number[]> {
    return (driver.executeLambda(async (txn: TransactionExecutor) => {
            let listOfNumbers: number[] = [1, 2, 3, 4, 5];
            // Updating QLDB
            await txn.execute("UPDATE ExampleTable SET ExampleList = ?",
 listOfNumbers);

            // Fetching from QLDB
            const resultList: dom.Value[] = (await txn.execute("SELECT VALUE
 ExampleList FROM ExampleTable")).getResultList();

            // Assume there is only one document in ExampleTable
            const ionValue: dom.Value = resultList[0];
            // Get Ion List
            const ionList: dom.Value[] = ionValue.elements();
            // Iterate through the Ion List to map it to a JavaScript Array
            let intList: number[] = [];
            ionList.forEach(item => {
                // Transforming Ion to a TypeScript Number
                const intValue: number = item.numberValue();
                intList.push(intValue);
            });
            listOfNumbers = intList;
            return listOfNumbers;
        })
    );
}
```

Python

```
def update_and_query_ion_list(txn):
    # QLDB can take in a Python list
    a_list = list()
    for i in range(0, 5):
        a_list.append(i)

    # Insertion into QLDB
    txn.execute_statement("UPDATE ExampleTable SET ExampleList = ?", a_list)

    # Fetching from QLDB
    cursor = txn.execute_statement("SELECT VALUE ExampleList FROM ExampleTable")

    # Assume there is only one document in ExampleTable
    for ion_value in cursor:
        # Ion Python blob is a child class of Python list
        a_list = ion_value

    # example_list is now the value fetched from QLDB
    return a_list


example_list = driver.execute_lambda(lambda txn: update_and_query_ion_list(txn))
```

Struct

Dans QLDB,`struct` les types de données sont particulièrement uniques par rapport aux autres types d'ions. Les documents de haut niveau que vous insérez dans un tableau doivent être du`struct` type. Un champ de document peut également contenir un champ imbriqué`struct`.

Par souci de simplicité, les exemples suivants définissent un document contenant uniquement`ExampleInt` les champs`ExampleString` et.

Java

```
class ExampleStruct {
    public String exampleString;
    public int exampleInt;

    public ExampleStruct(String exampleString, int exampleInt) {
```

```
        this.exampleString = exampleString;
        this.exampleInt = exampleInt;
    }
}

// Instantiate an IonSystem from the Ion library
IonSystem ionSystem = IonSystemBuilder.standard().build();

ExampleStruct examplePojo = driver.execute((txn) -> {
    // Transforming a POJO to Ion
    ExampleStruct aPojo = new ExampleStruct("Hello world!", 256);
    // Create an empty Ion struct
    IonStruct ionStruct = ionSystem.newEmptyStruct();
    // Map the fields of the POJO to Ion values and put them in the Ion struct
    ionStruct.add("ExampleString", ionSystem.newString(aPojo.exampleString));
    ionStruct.add("ExampleInt", ionSystem.newInt(aPojo.exampleInt));

    // Insertion into QLDB
    txn.execute("INSERT INTO ExampleTable ?", ionStruct);

    // Fetching from QLDB
    Result result = txn.execute("SELECT * from ExampleTable");
    // Assume there is only one document in ExampleTable
    for (IonValue ionValue : result)
    {
        // Map the fields of the Ion struct to Java values and construct a new POJO
        ionStruct = (IonStruct)ionValue;
        IonString exampleString = (IonString)ionStruct.get("ExampleString");
        IonInt exampleInt = (IonInt)ionStruct.get("ExampleInt");
        aPojo = new ExampleStruct(exampleString.stringValue(),
 exampleInt.intValue());
    }

    // examplePojo is now the document fetched from QLDB
    return aPojo;
});
```

Vous pouvez également utiliser la bibliothèque Jackson pour mapper les types de données depuis et vers Ion. La bibliothèque prend en charge les autres types de données Ion, mais cet exemple se concentre sur le`struct` type.

```
class ExampleStruct {
    public String exampleString;
```

```
    public int exampleInt;

    @JsonCreator
    public ExampleStruct(@JsonProperty("ExampleString") String exampleString,
                         @JsonProperty("ExampleInt") int exampleInt) {
        this.exampleString = exampleString;
        this.exampleInt = exampleInt;
    }

    @JsonProperty("ExampleString")
    public String getExampleString() {
        return this.exampleString;
    }

    @JsonProperty("ExampleInt")
    public int getExampleInt() {
        return this.exampleInt;
    }
}

// Instantiate an IonSystem from the Ion library
IonSystem ionSystem = IonSystemBuilder.standard().build();
// Instantiate an IonObjectMapper from the Jackson library
IonObjectMapper ionMapper = new IonValueMapper(ionSystem);

ExampleStruct examplePojo = driver.execute((txn) -> {
    // Transforming a POJO to Ion
    ExampleStruct aPojo = new ExampleStruct("Hello world!", 256);
    IonValue ionStruct;
    try  {
        // Use the mapper to convert Java objects into Ion
        ionStruct = ionMapper.writeValueAsIonValue(aPojo);
    } catch (IOException e) {
        // Wrap the exception and throw it for the sake of simplicity in this
 example
        throw new RuntimeException(e);
    }

    // Insertion into QLDB
    txn.execute("INSERT INTO ExampleTable ?", ionStruct);

    // Fetching from QLDB
    Result result = txn.execute("SELECT * from ExampleTable");
    // Assume there is only one document in ExampleTable
```

```
    for (IonValue ionValue : result)
    {
        // Use the mapper to convert Ion to Java objects
        try {
            aPojo = ionMapper.readValue(ionValue, ExampleStruct.class);
        } catch (IOException e) {
            // Wrap the exception and throw it for the sake of simplicity in this
 example
            throw new RuntimeException(e);
        }
    }

    // examplePojo is now the document fetched from QLDB
    return aPojo;
});
```

.NET

Utilisez la bibliothèque Amazon.Qldb.Driver.Serialization pour mapper les types de données C# natifs vers et depuis Ion. La bibliothèque prend en charge les autres types de données Ion, mais cet exemple se concentre sur le`struct` type.

```
using Amazon.QLDB.Driver.Generic;
using Amazon.QLDB.Driver.Serialization;
...

IAsyncQldbDriver driver = AsyncQldbDriver.Builder()
    .WithLedger("vehicle-registration")
    // Add Serialization library
    .WithSerializer(new ObjectSerializer())
    .Build();

// Creating a C# POCO.
ExampleStruct exampleStruct = new ExampleStruct
{
    ExampleString = "Hello world!",
    ExampleInt = 256
};

IAsyncResult<ExampleStruct> selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
```

```
    await txn.Execute(txn.Query<Document>("UPDATE ExampleTable SET ExampleStruct
 = ?", exampleStruct));

    // Fetching from QLDB.
    return await txn.Execute(txn.Query<ExampleStruct>("SELECT VALUE ExampleStruct
 from ExampleTable"));
});

await foreach (ExampleStruct row in selectResult)
{
    Console.WriteLine(row.ExampleString);
    Console.WriteLine(row.ExampleInt);
}
```

> **Note**
>
> Pour convertir en code synchrone, supprimez les`async` mots-clés`await` et et remplacez le`IAsyncResult` type par`IResult`.

Vous pouvez également utiliser Amazon. IonDotnetBibliothèque .Builders pour traiter les types de données Ion.

```
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;
...

IValueFactory valueFactory = new ValueFactory();

// Creating Ion struct.
IIonValue ionStruct = valueFactory.NewEmptyStruct();
ionStruct.SetField("ExampleString", valueFactory.NewString("Hello world!"));
ionStruct.SetField("ExampleInt", valueFactory.NewInt(256));

IAsyncResult selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
    await txn.Execute("UPDATE ExampleTable SET ExampleStruct = ?", ionStruct);

    // Fetching from QLDB.
    return await txn.Execute("SELECT VALUE ExampleStruct from ExampleTable");
});
```

```
string retrievedString = null;
int? retrievedInt = null;

await foreach (IIonValue ionValue in selectResult)
{
    retrievedString = ionValue.GetField("ExampleString").StringValue;
    retrievedInt = ionValue.GetField("ExampleInt").IntValue;
}
```

Go

```
exampleStruct, err := driver.Execute(context.Background(), func(txn
 qldbdriver.Transaction) (interface{}, error) {
 aStruct := map[string]interface{} {
  "ExampleString": "Hello World!",
  "ExampleInt": 256,
 }

 // Insertion into QLDB
 _, err = txn.Execute("UPDATE ExampleTable SET ExampleStruct = ?", aStruct)
 if err != nil {
  return nil, err
 }

 // Fetching from QLDB
 result, err := txn.Execute("SELECT VALUE ExampleStruct FROM ExampleTable")
 if err != nil {
  return nil, err
 }

 // Assume there is only one document in ExampleTable
 if result.Next(txn) {
  var decodedResult map[string]interface{}
  err := ion.Unmarshal(result.GetCurrentData(), &decodedResult)
  if err != nil {
   return nil, err
  }

  // exampleStruct is now the value fetched from QLDB
  return decodedResult, nil
 }
 return nil, result.Err()
})
```

Node.js

```
async function queryIonStruct(driver: QldbDriver): Promise<any> {
    let exampleStruct: any = {stringValue: "Hello World!", intValue: 256};
    return (driver.executeLambda(async (txn: TransactionExecutor) => {
            // Inserting into QLDB
            await txn.execute("INSERT INTO ExampleTable ?", exampleStruct);

            // Fetching from QLDB
            const resultList: dom.Value[] = (await txn.execute("SELECT * FROM
 ExampleTable")).getResultList();

            // Assume there is only one document in ExampleTable
            const ionValue: dom.Value = resultList[0];
            // We can get all the keys of Ion struct and their associated values
            const ionFieldNames: string[] = ionValue.fieldNames();

            // Getting key and value of Ion struct to TypeScript String and Number
            const nativeStringVal: string =
 ionValue.get(ionFieldNames[0]).stringValue();
            const nativeIntVal: number =
 ionValue.get(ionFieldNames[1]).numberValue();
            // Alternatively, we can access to Ion struct fields, using their
 literal field names:
            //   const nativeStringVal = ionValue.get("stringValue").stringValue();
            //   const nativeIntVal = ionValue.get("intValue").numberValue();

            exampleStruct = {[ionFieldNames[0]]: nativeStringVal,
 [ionFieldNames[1]]: nativeIntVal};
            return exampleStruct;
        })
    );
}
```

Python

```
def update_and_query_ion_struct(txn):
    # QLDB can take in a Python struct
    a_struct = {"ExampleString": "Hello world!", "ExampleInt": 256}

    # Insertion into QLDB
    txn.execute_statement("UPDATE ExampleTable SET ExampleStruct = ?", a_struct)
```

```
    # Fetching from QLDB
    cursor = txn.execute_statement("SELECT VALUE ExampleStruct FROM ExampleTable")

    # Assume there is only one document in ExampleTable
    for ion_value in cursor:
        # Ion Python struct is a child class of Python struct
        a_struct = ion_value

    # example_struct is now the value fetched from QLDB
    return a_struct


example_struct = driver.execute_lambda(lambda txn: update_and_query_ion_struct(txn))
```

Valeurs nulles et types dynamiques

QLDB prend en charge le contenu ouvert et n'applique pas les définitions de schéma ou de type de données pour les champs du document. Vous pouvez également stocker les valeurs nulles d'Ion dans un document QLDB. Tous les exemples précédents supposent que chaque type de données renvoyé est connu et n'est pas nul. Les exemples suivants montrent comment utiliser Ion lorsque le type de données n'est pas connu ou est peut-être nul.

Java

```
// Empty variables
String exampleString = null;
Integer exampleInt = null;

// Assume ionValue is some queried data from QLDB
IonValue ionValue = null;

// Check the value type and assign it to the variable if it is not null
if (ionValue.getType() == IonType.STRING) {
    if (ionValue.isNullValue()) {
        exampleString = null;
    } else {
        exampleString = ((IonString)ionValue).stringValue();
    }
} else if (ionValue.getType() == IonType.INT) {
    if (ionValue.isNullValue()) {
        exampleInt = null;
```

```
    } else {
        exampleInt = ((IonInt)ionValue).intValue();
    }
};

// Creating null values
IonSystem ionSystem = IonSystemBuilder.standard().build();

// A null value still has an Ion type
IonString ionString;
if (exampleString == null) {
    // Specifically a null string
    ionString = ionSystem.newNullString();
} else {
    ionString = ionSystem.newString(exampleString);
}

IonInt ionInt;
if (exampleInt == null) {
    // Specifically a null int
    ionInt = ionSystem.newNullInt();
} else {
    ionInt = ionSystem.newInt(exampleInt);
}

// Special case regarding null!
// There is a generic null type which has Ion type 'Null'.
// The above null values still have the types 'String' and 'Int'
IonValue specialNull = ionSystem.newNull();
if (specialNull.getType() == IonType.NULL) {
    // This is true!
}
if (specialNull.isNullValue()) {
    // This is also true!
}
if (specialNull.getType() == IonType.STRING || specialNull.getType() == IonType.INT)
 {
    // This is false!
}
```

.NET

```
// Empty variables.
```

```
string exampleString = null;
int? exampleInt = null;

// Assume ionValue is some queried data from QLDB.
IIonValue ionValue;

if (ionValue.Type() == IonType.String)
{
    exampleString = ionValue.StringValue;
}
else if (ionValue.Type() == IonType.Int)
{
    if (ionValue.IsNull)
    {
        exampleInt = null;
    }
    else
    {
        exampleInt = ionValue.IntValue;
    }
};

// Creating null values.
IValueFactory valueFactory = new ValueFactory();

// A null value still has an Ion type.
IIonValue ionString = valueFactory.NewString(exampleString);

IIonValue ionInt;
if (exampleInt == null)
{
    // Specifically a null int.
    ionInt = valueFactory.NewNullInt();
}
else
{
    ionInt = valueFactory.NewInt(exampleInt.Value);
}

// Special case regarding null!
// There is a generic null type which has Ion type 'Null'.
IIonValue specialNull = valueFactory.NewNull();
if (specialNull.Type() == IonType.Null) {
    // This is true!
```

```
}
if (specialNull.IsNull) {
    // This is also true!
}
```

Go

Limites de triage

Dans Go,`nil` les valeurs ne conservent pas leur type lorsque vous les assemblez puis que vous les désassemblez. Une`nil` valeur passe à Ion null, mais Ion null la désorganise à une valeur nulle plutôt qu'à zéro`nil`.

```
ionNull, err := ion.MarshalText(nil) // ionNull is set to ion null
if err != nil {
    return
}

var result int
err = ion.Unmarshal(ionNull, &result) // result unmarshals to 0
if err != nil {
    return
}
```

Node.js

```
// Empty variables
let exampleString: string;
let exampleInt: number;

// Assume ionValue is some queried data from QLDB
// Check the value type and assign it to the variable if it is not null
if (ionValue.getType() === IonTypes.STRING) {
    if (ionValue.isNull()) {
        exampleString = null;
    } else {
        exampleString = ionValue.stringValue();
    }
} else if (ionValue.getType() === IonTypes.INT) {
    if (ionValue.isNull()) {
        exampleInt = null;
    } else {
        exampleInt = ionValue.numberValue();
```

```
    }
}

// Creating null values
if (exampleString === null) {
    ionString = dom.load('null.string');
} else {
    ionString = dom.load.of(exampleString);
}

if (exampleInt === null) {
    ionInt = dom.load('null.int');
} else {
    ionInt = dom.load.of(exampleInt);
}

// Special case regarding null!
// There is a generic null type which has Ion type 'Null'.
// The above null values still have the types 'String' and 'Int'
specialNull: dom.Value = dom.load("null.null");
if (specialNull.getType() === IonType.NULL) {
    // This is true!
}
if (specialNull.getType() === IonType.STRING || specialNull.getType() ===
 IonType.INT) {
    // This is false!
}
```

Python

```
# Empty variables
example_string = None
example_int = None

# Assume ion_value is some queried data from QLDB
# Check the value type and assign it to the variable if it is not null
if ion_value.ion_type == IonType.STRING:
    if isinstance(ion_value, IonPyNull):
        example_string = None
    else:
        example_string = ion_value
elif ion_value.ion_type == IonType.INT:
    if isinstance(ion_value, IonPyNull):
```

```
        example_int = None
    else:
        example_int = ion_value

# Creating Ion null values
if example_string is None:
    # Specifically a null string
    ion_string = loads("null.string")
else:
    # QLDB can take in Python string
    ion_string = example_string
if example_int is None:
    # Specifically a null int
    ion_int = loads("null.int")
else:
    # QLDB can take in Python int
    ion_int = example_int


# Special case regarding null!
# There is a generic null type which has Ion type 'Null'.
# The above null values still have the types 'String' and 'Int'
special_null = loads("null.null")
if special_null.ion_type == IonType.NULL:
    # This is true!
if special_null.ion_type == IonType.STRING or special_null.ion_type == IonType.INT:
    # This is false!
```

Conversion vers le format JSON

Si votre application nécessite la compatibilité JSON, vous pouvez rétroconvertir les données Amazon Ion en JSON. Cependant, la conversion d'ions en JSON entraîne des pertes dans certains cas où vos données utilisent des types d'ions riches qui n'existent pas dans JSON.

Pour plus de détails sur les règles de conversion d'ion en JSON, consultez la section Conversion descendante en JSON dans le livre de recettes Amazon Ion.

Utilisation des données et de l'historique dans Amazon QLDB

Les rubriques suivantes fournissent des exemples de base d'instructions de création, lecture, mise à jour et suppression (CRUD). Vous pouvez exécuter ces instructions manuellement à l'aide de l'éditeur PartiQL de la console QLDB ou du shell QLDB. Ce guide vous explique également comment QLDB gère vos données lorsque vous apportez des modifications à votre registre.

QLDB prend en charge le langage de requête PartiQL.

Pour des exemples de code montrant comment exécuter par programmation des instructions similaires à l'aide du pilote QLDB, consultez les didacticiels dansDémarrage.

> **Tip**
>
> Vous trouverez ci-dessous un bref résumé des conseils et des meilleures pratiques pour utiliser PartiQL dans QLDB :
>
> - Comprenez les limites de simultanéité et de transaction : toutes les déclarations, y compris les`SELECT` requêtes, sont soumises à des conflits et à des limites de transaction optimistes (OCC), y compris un délai d'expiration de 30 secondes pour les transactions.
> - Utilisez des index : utilisez des index à cardinalité élevée et exécutez des requêtes ciblées pour optimiser vos déclarations et éviter de scanner des tableaux complets. Pour en savoir plus, consultez Optimisation des performances des données.
> - Utiliser des prédicats d'égalité : les recherches indexées nécessitent un opérateur d'égalité (`=` ou`IN`). Les opérateurs d'inégalité (`<>`,`LIKE`,,`BETWEEN`) ne sont pas éligibles aux recherches indexées et donnent lieu à des analyses de tableaux complets.
> - Utiliser uniquement les jointures internes : QLDB ne prend en charge que les jointures internes. Il est recommandé de joindre des champs indexés pour chaque table que vous joignez. Choisissez des indices de cardinalité élevée pour les critères de jointure et les prédicats d'égalité.

Rubriques

- Création de tableaux avec index et insertion de documents
- Interrogation de données

- Interroger les métadonnées d'un document
- Utilisation de la clause BY pour interroger l'ID du document
- Mise à jour et suppression de documents
- Consultation de l'historique des révisions
- Rédaction de révisions de documents
- Optimisation des performances des données
- Obtention des statistiques d'instruction PartiQL
- Interrogation du catalogue système
- Gestion des tables
- Gestion des index
- Identifiants uniques dans Amazon QLDB

Création de tableaux avec index et insertion de documents

Après avoir créé un registre Amazon QLDB, la première étape consiste à créer un tableau contenant uneCREATE TABLE déclaration de base. Les tableaux se composent deDocuments QLDB, qui sont des ensembles de données`struct` au format Amazon Ion.

Rubriques

- Création de tables et d'index
- Insertion de documents

Création de tables et d'index

Les tableaux ont des noms simples qui distinguent les majuscules des minuscules, sans espaces de noms. QLDB prend en charge le contenu ouvert et n'applique pas de schéma. Vous ne définissez donc pas d'attributs ou de types de données lors de la création de tables.

```
CREATE TABLE VehicleRegistration
```

```
CREATE TABLE Vehicle
```

Une`CREATE TABLE` instruction renvoie l'ID attribué par le système à la nouvelle table. Tous les identifiants attribués par le système dans QLDB sont des identifiants uniques universels (UUID) qui sont chacun représentés dans une chaîne codée en Base62.

> **Note**
>
> Vous pouvez éventuellement définir des balises pour une ressource de table lors de la création de la table. Pour savoir comment procéder, veuillez consulter la section Baliser les tables lors de la création.

Vous pouvez également créer des index sur des tables afin d'optimiser les performances des tables.

```
CREATE INDEX ON VehicleRegistration (VIN)
```

```
CREATE INDEX ON VehicleRegistration (LicensePlateNumber)
```

```
CREATE INDEX ON Vehicle (VIN)
```

> **Important**
>
> QLDB a besoin d'un index pour rechercher efficacement un document. Sans index, QLDB doit effectuer une analyse complète de la table lors de la lecture de documents. Cela peut entraîner des problèmes de performances sur des tables de grande taille, notamment des conflits de simultanéité et des délais de transaction.
> Pour éviter de scanner des tables, vous devez exécuter des instructions avec une clause de`WHERE` prédicat à l'aide d'un opérateur d'égalité (`=`ou`IN`) sur un champ indexé ou un identifiant de document. Pour plus d'informations, veuillez consulter Optimisation des performances des données.

Tenez compte des contraintes suivantes lors de la création d'index :

- Un index ne peut être créé que sur un seul champ de niveau supérieur. Les index composites, imbriqués, uniques et basés sur des fonctions ne sont pas pris en charge.
- Vous pouvez créer un index sur tous les types de données Ion, y compris`list` et`struct`. Toutefois, vous ne pouvez effectuer la recherche indexée que par égalité de la valeur totale des

ions, quel que soit le type d'ion. Par exemple, lorsque vous utilisez un`list` type comme index, vous ne pouvez pas effectuer de recherche indexée par un élément de la liste.

- Les performances des requêtes ne sont améliorées que lorsque vous utilisez un prédicat d'égalité ; par exemple,`WHERE indexedField = 123` ou`WHERE indexedField IN (456, 789)`.

  QLDB ne respecte pas les inégalités dans les prédicats de requête. Par conséquent, les analyses filtrées par plage de valeurs ne sont pas mises en œuvre.
- Les noms de champs indexés sont sensibles à la casse et peuvent contenir un maximum de 128 caractères.
- La création d'index dans QLDB est asynchrone. Le temps nécessaire à la création d'un index sur une table non vide varie en fonction de la taille de la table. Pour plus d'informations, veuillez consulter Gestion des index.

Insertion de documents

Vous pouvez ensuite insérer des documents dans vos tableaux. Les documents QLDB sont stockés au format Amazon Ion. LesINSERT instructions partiQL suivantes incluent un sous-ensemble des exemples de données d'immatriculation des véhicules utilisés dansMise en route avec la console Amazon QLDB.

```
INSERT INTO VehicleRegistration
<< {
    'VIN' : '1N4AL11D75C109151',
    'LicensePlateNumber' : 'LEWISR261LL',
    'State' : 'WA',
    'City' : 'Seattle',
    'PendingPenaltyTicketAmount' : 90.25,
    'ValidFromDate' : `2017-08-21T`,
    'ValidToDate' : `2020-05-11T`,
    'Owners' : {
        'PrimaryOwner' : { 'PersonId' : '294jJ3YUoH1IEEm8GSabOs' },
        'SecondaryOwners' : [ { 'PersonId' : '5Ufgdlnj06gF5CWcOIu64s' } ]
    }
},
{
    'VIN' : 'KM8SRDHF6EU074761',
    'LicensePlateNumber' : 'CA762X',
    'State' : 'WA',
    'City' : 'Kent',
```

```
    'PendingPenaltyTicketAmount' : 130.75,
    'ValidFromDate' : `2017-09-14T`,
    'ValidToDate' : `2020-06-25T`,
    'Owners' : {
        'PrimaryOwner' : { 'PersonId': 'IN7MvYtUjkp1GMZu0F6CG9' },
        'SecondaryOwners' : []
    }
} >>
```

```
INSERT INTO Vehicle
<< {
    'VIN' : '1N4AL11D75C109151',
    'Type' : 'Sedan',
    'Year' : 2011,
    'Make' : 'Audi',
    'Model' : 'A5',
    'Color' : 'Silver'
} ,
{
    'VIN' : 'KM8SRDHF6EU074761',
    'Type' : 'Sedan',
    'Year' : 2015,
    'Make' : 'Tesla',
    'Model' : 'Model S',
    'Color' : 'Blue'
} >>
```

Syntaxe et sémantique de PartiQL

- Les noms des champs sont placés entre guillemets simples (`'...'`).
- Les valeurs de chaîne sont également placées entre guillemets simples (`'...'`).
- Les horodatages sont placés entre crochets inverses (`` `...` ``). Les backticks peuvent être utilisés pour désigner n'importe quel littéral ionique.
- Les nombres entiers et décimaux sont des valeurs littérales qui n'ont pas besoin d'être indiquées.

Pour plus de détails sur la syntaxe et la sémantique de PartiQL, consultezInterroger Ion avec PartiQL dans Amazon QLDB.

Une`INSERT` instruction crée la révision initiale d'un document dont le numéro de version est zéro. Pour identifier de manière unique chaque document, QLDB attribue un identifiant de document dans le cadre des métadonnées. Les instructions d'insertion renvoient l'ID de chaque document inséré.

> **Important**
>
> Comme QLDB n'applique pas de schéma, vous pouvez insérer le même document dans un tableau plusieurs fois. Chaque instruction d'insertion valide une entrée de document distincte dans le journal, et QLDB attribue à chaque document un identifiant unique.

Pour savoir comment interroger les documents que vous avez insérés dans votre tableau, passez àInterrogation de données.

Interrogation de données

La vue utilisateur renvoie uniquement la dernière révision non supprimée de vos données utilisateur. Il s'agit de la vue par défaut dans Amazon QLDB. Cela signifie qu'aucun qualificatif spécial n'est nécessaire lorsque vous souhaitez interroger uniquement vos données.

Pour plus d'informations sur la syntaxe et les paramètres des exemples de requêtes suivants, consultezSELECT la référence Amazon QLDB PartiQL.

Rubriques

- Requête de base
- Projections et filtres
- Jointures
- Données imbriquées

Requête de base

Les tables`SELECT` de base renvoient les documents que vous avez insérés dans la table.

> **Warning**
>
> Lorsque vous exécutez une requête dans QLDB sans recherche indexée, elle appelle une analyse complète de la table. PartiQL prend en charge de telles requêtes car il est compatible

avec SQL. Toutefois, n'exécutez pas d'analyses de tables pour des cas d'utilisation en production dans QLDB. Les analyses de tables peuvent entraîner des problèmes de performances sur des tables de grande taille, notamment des conflits de simultanéité et des délais de transaction.
Pour éviter de scanner des tables, vous devez exécuter des instructions avec une clause de`WHERE` prédicat à l'aide d'un opérateur d'égalité sur un champ indexé ou un identifiant de document, par exemple,`WHERE indexedField = 123` ou`WHERE indexedField IN (456, 789)`. Pour plus d'informations, veuillez consulter Optimisation des performances des données.

Les requêtes suivantes affichent les résultats des documents d'immatriculation du véhicule que vous avez précédemment insérésCréation de tableaux avec index et insertion de documents. L'ordre des résultats n'est pas spécifique et peut varier pour chaque`SELECT` requête. Vous ne devez pas vous fier à l'ordre des résultats pour aucune requête dans QLDB.

```
SELECT * FROM VehicleRegistration
WHERE LicensePlateNumber IN ('LEWISR261LL', 'CA762X')
```

```
{
    VIN: "1N4AL11D75C109151",
    LicensePlateNumber: "LEWISR261LL",
    State: "WA",
    City: "Seattle",
    PendingPenaltyTicketAmount: 90.25,
    ValidFromDate: 2017-08-21T,
    ValidToDate: 2020-05-11T,
    Owners: {
        PrimaryOwner: { PersonId: "294jJ3YUoH1IEEm8GSabOs" },
        SecondaryOwners: [{ PersonId: "5Ufgdlnj06gF5CWcOIu64s" }]
    }
},
{
    VIN: "KM8SRDHF6EU074761",
    LicensePlateNumber: "CA762X",
    State: "WA",
    City: "Kent",
    PendingPenaltyTicketAmount: 130.75,
    ValidFromDate: 2017-09-14T,
    ValidToDate: 2020-06-25T,
```

```
    Owners: {
        PrimaryOwner: { PersonId: "IN7MvYtUjkp1GMZu0F6CG9" },
        SecondaryOwners: []
    }
}
```

```
SELECT * FROM Vehicle
WHERE VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

```
{
    VIN: "1N4AL11D75C109151",
    Type: "Sedan",
    Year: 2011,
    Make: "Audi",
    Model: "A5",
    Color: "Silver"
},
{
    VIN: "KM8SRDHF6EU074761",
    Type: "Sedan",
    Year: 2015,
    Make: "Tesla",
    Model: "Model S",
    Color: "Blue"
}
```

Important

Dans PartiQL, vous utilisez des guillemets simples pour désigner des chaînes en langage de manipulation de données (DML) ou dans des instructions de requête. Mais la console QLDB et le shell QLDB renvoient les résultats des requêtes au format texte Amazon Ion. Vous pouvez donc voir des chaînes entre guillemets doubles.
Cette syntaxe permet au langage de requête PartiQL de maintenir la compatibilité SQL et au format de texte Amazon Ion de maintenir la compatibilité JSON.

Projections et filtres

Vous pouvez faire des projections (cibléesSELECT) et d'autres filtres standard (WHEREclauses). La requête suivante renvoie un sous-ensemble de champs de document à partir de laVehicleRegistration table. Il filtre les véhicules répondant aux critères suivants :

- Filtre de chaîne — Il est enregistré à Seattle.
- Filtre décimal : le montant du ticket de pénalité en attente est inférieur à100.0.
- Filtre par date : sa date d'enregistrement est valide à compter du 4 septembre 2019.

```
SELECT r.VIN, r.PendingPenaltyTicketAmount, r.Owners
FROM VehicleRegistration AS r
WHERE r.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
AND r.City = 'Seattle' --string
AND r.PendingPenaltyTicketAmount < 100.0 --decimal
AND r.ValidToDate >= `2019-09-04T` --timestamp with day precision
```

```
{
    VIN: "1N4AL11D75C109151",
    PendingPenaltyTicketAmount: 90.25,
    Owners: {
        PrimaryOwner: { PersonId: "294jJ3YUoH1IEEm8GSabOs" },
        SecondaryOwners: [{ PersonId: "5Ufgdlnj06gF5CWcOIu64s" }]
    }
}
```

Jointures

Vous pouvez également écrire des requêtes de jointure internes. L'exemple suivant montre une requête de jointure interne implicite qui renvoie tous les documents d'immatriculation ainsi que les attributs des véhicules immatriculés.

```
SELECT * FROM VehicleRegistration AS r, Vehicle AS v
WHERE r.VIN = v.VIN
AND r.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

```
{
    VIN: "1N4AL11D75C109151",
```

```
    LicensePlateNumber: "LEWISR261LL",
    State: "WA",
    City: "Seattle",
    PendingPenaltyTicketAmount: 90.25,
    ValidFromDate: 2017-08-21T,
    ValidToDate: 2020-05-11T,
    Owners: {
        PrimaryOwner: { PersonId: "294jJ3YUoH1IEEm8GSabOs" },
        SecondaryOwners: [{ PersonId: "5Ufgdlnj06gF5CWcOIu64s" }]
    },
    Type: "Sedan",
    Year: 2011,
    Make: "Audi",
    Model: "A5",
    Color: "Silver"
},
{
    VIN: "KM8SRDHF6EU074761",
    LicensePlateNumber: "CA762X",
    State: "WA",
    City: "Kent",
    PendingPenaltyTicketAmount: 130.75,
    ValidFromDate: 2017-09-14T,
    ValidToDate: 2020-06-25T,
    Owners: {
        PrimaryOwner: { PersonId: "IN7MvYtUjkp1GMZu0F6CG9" },
        SecondaryOwners: []
    },
    Type: "Sedan",
    Year: 2015,
    Make: "Tesla",
    Model: "Model S",
    Color: "Blue"
}
```

Vous pouvez également écrire la même requête de jointure interne dans la syntaxe explicite suivante.

```
SELECT * FROM VehicleRegistration AS r INNER JOIN Vehicle AS v
ON r.VIN = v.VIN
WHERE r.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

Données imbriquées

Vous pouvez utiliser PartiQL dans QLDB pour interroger des données imbriquées dans des documents. L'exemple suivant montre une sous-requête corrélée qui aplatit les données imbriquées. Le@ personnage est ici techniquement facultatif. Mais cela indique explicitement que vous souhaitez que la`Owners` structure soit intégrée`VehicleRegistration`, et non qu'une collection différente soit nommée`Owners` (s'il en existe une).

```
SELECT
    r.VIN,
    o.SecondaryOwners
FROM
    VehicleRegistration AS r, @r.Owners AS o
WHERE
    r.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

```
{
    VIN: "1N4AL11D75C109151",
    SecondaryOwners: [{ PersonId: "5Ufgdlnj06gF5CWcOIu64s" }]
},
{
    VIN: "KM8SRDHF6EU074761",
    SecondaryOwners: []
}
```

Ce qui suit montre une sous-requête de la`SELECT` liste qui projette des données imbriquées, en plus d'une jointure interne.

```
SELECT
    v.Make,
    v.Model,
    (SELECT VALUE o.PrimaryOwner.PersonId FROM @r.Owners AS o) AS PrimaryOwner
FROM
    VehicleRegistration AS r, Vehicle AS v
WHERE
    r.VIN = v.VIN AND r.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

```
{
    Make: "Audi",
    Model: "A5",
    PrimaryOwner: ["294jJ3YUoH1IEEm8GSabOs"]
```

```
},
{
    Make: "Tesla",
    Model: "Model S",
    PrimaryOwner: ["IN7MvYtUjkp1GMZu0F6CG9"]
}
```

La requête suivante renvoie le numéro d'index`PersonId` et (ordinal) de chaque personne figurant dans la`Owners.SecondaryOwners` liste d'un`VehicleRegistration` document.

```
SELECT s.PersonId, owner_idx
FROM VehicleRegistration AS r, @r.Owners.SecondaryOwners AS s AT owner_idx
WHERE r.VIN = '1N4AL11D75C109151'
```

```
{
    PersonId: "5Ufgdlnj06gF5CWcOIu64s",
    owner_idx: 0
}
```

Pour savoir comment interroger les métadonnées de vos documents, passez àInterroger les métadonnées d'un document.

Interroger les métadonnées d'un document

Une`INSERT` instruction crée la révision initiale d'un document dont le numéro de version est zéro. Pour identifier de manière unique chaque document, Amazon QLDB attribue un identifiant de document dans le cadre des métadonnées.

Outre l'identifiant du document et le numéro de version, QLDB stocke d'autres métadonnées générées par le système pour chaque document dans un tableau. Ces métadonnées incluent les informations de transaction, les attributs du journal et la valeur de hachage du document.

Tous les identifiants attribués par le système sont des identifiants uniques universels (UUID) qui sont chacun représentés dans une chaîne codée en Base62. Pour plus d'informations, veuillez consulter Identifiants uniques dans Amazon QLDB.

Rubriques

Point de vue engagé

Vous pouvez accéder aux métadonnées du document en interrogeant la vue validée. Cette vue renvoie les documents de la table définie par le système qui correspond directement à votre table utilisateur. Il inclut la dernière révision validée et non supprimée de vos données et des métadonnées générées par le système. Pour interroger cette vue, ajoutez le préfixe`_ql_committed_` au nom de la table dans votre requête. (Le préfixe`_ql_` est réservé dans QLDB aux objets système.)

```
SELECT * FROM _ql_committed_VehicleRegistration AS r
WHERE r.data.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

À l'aide des données précédemment inséréesCréation de tableaux avec index et insertion de documents, la sortie de cette requête affiche le contenu système de la dernière révision de chaque document non supprimé. Les métadonnées du document système sont imbriquées dans le`metadata` champ et vos données utilisateur sont imbriquées dans le`data` champ.

```
{
    blockAddress:{
        strandId:"JdxjkR9bSYB5jMHWcI464T",
        sequenceNo:14
    },
    hash:{{wCsmM6qD4STxz0WYmE+47nZvWtcCz9D6zNtCiM5GoWg=}},
    data:{
        VIN: "1N4AL11D75C109151",
        LicensePlateNumber: "LEWISR261LL",
        State: "WA",
        City: "Seattle",
        PendingPenaltyTicketAmount: 90.25,
        ValidFromDate: 2017-08-21T,
        ValidToDate: 2020-05-11T,
        Owners: {
            PrimaryOwner: { PersonId: "294jJ3YUoH1IEEm8GSabOs" },
            SecondaryOwners: [{ PersonId: "5Ufgdlnj06gF5CWcOIu64s" }]
        }
    },
    metadata:{
        id:"3Qv67yjXEwB9SjmvkuG6Cp",
        version:0,
        txTime:2019-06-05T20:53:321d-3Z,
        txId:"HgXAkLjAtV0HQ4lNYdzX60"
    }
```

```
},
{
    blockAddress:{
        strandId:"JdxjkR9bSYB5jMHWcI464T",
        sequenceNo:14
    },
    hash:{{wPuwH60TtcCvg/23BFp+redRXuCALkbDihkEvCX22Jk=}},
    data:{
        VIN: "KM8SRDHF6EU074761",
        LicensePlateNumber: "CA762X",
        State: "WA",
        City: "Kent",
        PendingPenaltyTicketAmount: 130.75,
        ValidFromDate: 2017-09-14T,
        ValidToDate: 2020-06-25T,
        Owners: {
            PrimaryOwner: { PersonId: "IN7MvYtUjkp1GMZu0F6CG9" },
            SecondaryOwners: []
        }
    },
    metadata:{
        id:"JOzfB3lWqGU727mpPeWyxg",
        version:0,
        txTime:2019-06-05T20:53:321d-3Z,
        txId:"HgXAkLjAtV0HQ4lNYdzX60"
    }
}
```

Champs d'affichage validés

- `blockAddress`— L'emplacement du bloc dans le journal de votre grand livre où la révision du document a été validée. Une adresse, qui peut être utilisée pour la vérification cryptographique, comporte les deux champs suivants.
  - `strandId`— L'identifiant unique du volet du journal qui contient le bloc.
  - `sequenceNo`— Numéro d'index qui indique l'emplacement du bloc dans le brin.

> **Note**
>
> Dans cet exemple, les deux documents ont une`blockAddress` valeur identique et identique`sequenceNo`. Comme ces documents ont été insérés dans une seule transaction (et dans ce cas, dans une seule instruction), ils ont été validés dans le même bloc.

- `hash`— La valeur de hachage ionique SHA-256 qui représente de manière unique la révision du document. Le hachage couvre les révisions`data` et`metadata` les champs et peut être utilisé pour la vérification cryptographique.
- `data`— Les attributs des données utilisateur du document.

  Si vous rédigez une révision, cette`data` structure est remplacée par un`dataHash` champ dont la valeur est le hachage ionique de la`data` structure supprimée.
- `metadata`— Les attributs des métadonnées du document.
  - `id`— L'identifiant unique attribué par le système au document.
  - `version`— Numéro de version du document. Il s'agit d'un entier à base zéro qui augmente à chaque révision du document.
  - `txTime`— Horodatage auquel la révision du document a été validée dans le journal.
  - `txId`ID unique de la transaction ayant validé la révision du document.

Rejoindre les points de vue des utilisateurs et des utilisateurs

Vous pouvez écrire des requêtes qui joignent une table dans la vue validée à une table dans la vue utilisateur. Par exemple, vous souhaiterez peut-être joindre le document`id` d'une table à un champ défini par l'utilisateur d'une autre table.

La requête suivante joint deux tables nommées`DriversLicense` et`Person` sur leur`PersonId`id champ de document respectivement, en utilisant la vue validée pour cette dernière.

```
SELECT * FROM DriversLicense AS d INNER JOIN _ql_committed_Person AS p
ON d.PersonId = p.metadata.id
WHERE p.metadata.id = '1CWScY2qHYI9G88C2SjvtH'
```

Pour savoir comment interroger le champ d'identification du document dans la vue utilisateur par défaut, passez àUtilisation de la clause BY pour interroger l'ID du document.

Utilisation de la clause BY pour interroger l'ID du document

Bien que vous puissiez définir des champs destinés à être des identifiants uniques (par exemple, le VIN d'un véhicule), le véritable identifiant unique d'un document est le champ de`id` métadonnées, comme décrit dansInsertion de documents. C'est pourquoi vous pouvez utiliser le`id` champ pour créer des relations entre les tables.

Leid champ du document est directement accessible uniquement dans la vue validée, mais vous pouvez également le projeter dans la vue utilisateur par défaut à l'aide de laBY clause. Pour un exemple, consultez la requête suivante et ses résultats.

```
SELECT r_id, r.VIN, r.LicensePlateNumber, r.State, r.City, r.Owners
FROM VehicleRegistration AS r BY r_id
WHERE r_id = '3Qv67yjXEwB9SjmvkuG6Cp'
```

```
{
    r_id: "3Qv67yjXEwB9SjmvkuG6Cp",
    VIN: "1N4AL11D75C109151",
    LicensePlateNumber: "LEWISR261LL",
    State: "WA",
    City: "Seattle",
    Owners: {
        PrimaryOwner: { PersonId: "294jJ3YUoH1IEEm8GSabOs" },
        SecondaryOwners: [{ PersonId: "5Ufgdlnj06gF5CWcOIu64s" }]
    }
}
```

Dans cette requête,r_id se trouve un alias défini par l'utilisateur qui est déclaré dans laFROM clause, à l'aide duBY mot clé. Cetr_id alias est lié au champ deid métadonnées de chaque document du jeu de résultats de la requête. Vous pouvez utiliser cet alias dans laSELECT clause ainsi que dans laWHERE clause d'une requête dans la vue utilisateur.

Pour accéder à d'autres attributs de métadonnées, vous devez toutefois interroger la vue validée.

Adhésion via un numéro de document

Supposons que vous utilisiez le documentid d'une table comme clé étrangère dans un champ défini par l'utilisateur d'une autre table. Vous pouvez utiliser laBY clause pour écrire une requête de jointure interne pour les deux tables de ces champs (commeRejoindre les points de vue des utilisateurs et des utilisateurs dans la rubrique précédente).

L'exemple suivant joint deux tables nomméesDriversLicense etPerson sur leurPersonIdid champ de document, respectivement, à l'aide de laBY clause correspondant à ce dernier.

```
SELECT * FROM DriversLicense AS d INNER JOIN Person AS p BY pid
ON d.PersonId = pid
```

```
WHERE pid = '1CWScY2qHYI9G88C2SjvtH'
```

Pour savoir comment apporter des modifications à un document de votre tableau, passez àMise à jour et suppression de documents.

Mise à jour et suppression de documents

Dans Amazon QLDB, une révision de document est une structure Amazon Ion qui représente une version unique d'une séquence de documents identifiés par un identifiant de document unique. Chaque révision contient l'ensemble de données complet du document, y compris vos données utilisateur et les métadonnées générées par le système. Chaque révision est identifiée de manière unique par une combinaison de l'identifiant du document et d'un numéro de version en base zéro.

Lorsque vous mettez à jour un document, QLDB crée une nouvelle révision avec le même identifiant de document et un numéro de version incrémenté. Le cycle de vie d'un document prend fin lorsque vous le supprimez d'un tableau. Cela signifie qu'aucune révision de document avec le même identifiant de document ne peut être créée à nouveau.

Révision des documents

Par exemple, les instructions suivantes insèrent une nouvelle immatriculation du véhicule, mettent à jour la ville d'immatriculation, puis suppriment l'immatriculation. Il en résulte trois révisions d'un document.

```
INSERT INTO VehicleRegistration
{
    'VIN' : '1HVBBAANXWH544237',
    'LicensePlateNumber' : 'LS477D',
    'State' : 'WA',
    'City' : 'Tacoma',
    'PendingPenaltyTicketAmount' : 42.20,
    'ValidFromDate' : `2011-10-26T`,
    'ValidToDate' : `2023-09-25T`,
    'Owners' : {
        'PrimaryOwner' : { 'PersonId': 'KmA3XPKKFqYCP2zhR3d0Ho' },
        'SecondaryOwners' : []
    }
}
```

> **Note**
>
> Les instructions Insert et les autres instructions DML renvoient l'ID de chaque document concerné. Avant de continuer, enregistrez cet identifiant car vous en avez besoin pour la fonction historique de la rubrique suivante. Vous pouvez également trouver l'ID de document à l'aide de la requête suivante.
>
> ```
> SELECT r_id FROM VehicleRegistration AS r BY r_id
> WHERE r.VIN = '1HVBBAANXWH544237'
> ```

```
UPDATE VehicleRegistration AS r
SET r.City = 'Bellevue'
WHERE r.VIN = '1HVBBAANXWH544237'
```

```
DELETE FROM VehicleRegistration AS r
WHERE r.VIN = '1HVBBAANXWH544237'
```

Pour plus d'exemples et d'informations sur la syntaxe de ces instructions DML, consultezMISE A JOUR etDELETE consultez la référence Amazon QLDB PartiQL.

Pour insérer et supprimer des éléments spécifiques dans un document, vous pouvez utiliserUPDATE des instructions ou d'autres instructions DML qui commencent par leFROM mot clé. Pour plus d'informations et d'exemples, consultez laDE (INSÉRER, SUPPRIMER ou DÉFINIR) référence.

Une fois que vous avez supprimé un document, vous ne pouvez plus l'interroger dans les vues validées ou utilisateur. Pour savoir comment interroger l'historique des révisions de ce document à l'aide de la fonction d'historique intégrée, passez àConsultation de l'historique des révisions.

Consultation de l'historique des révisions

Amazon QLDB stocke l'historique complet de chaque document dans un tableau. Vous pouvez consulter les trois révisions du document d'immatriculation du véhicule que vous avez précédemment inséré, mis à jour et supprimé enMise à jour et suppression de documents interrogeant la fonction d'historique intégrée.

Rubriques

Fonction historique

La fonction d'historique de QLDB est une extension PartiQL qui renvoie les révisions à partir de la vue définie par le système de votre table. Il inclut donc à la fois vos données et les métadonnées associées dans le même schéma que la vue validée.

Syntaxe

```
SELECT * FROM history( table_name | 'table_id' [, `start-time` [, `end-time` ] ] ) AS h
[ WHERE h.metadata.id = 'id' ]
```

Arguments

nom_table | « ***id_table*** »

Le nom ou l'identifiant de la table. Un nom de table est un identifiant PartiQL que vous pouvez indiquer par des guillemets doubles ou sans guillemets. Un ID de table est une chaîne littérale qui doit être placée entre guillemets simples. Pour en savoir plus sur l'utilisation des ID de table, consultezConsultation de l'historique des tables inactives.

`***heure*** de début`, `***heure de fin***`

(Facultatif) Spécifie la période pendant laquelle les révisions étaient actives. Ces paramètres ne précisent pas la période pendant laquelle les révisions ont été validées dans le journal dans le cadre d'une transaction.

Les heures de début et de fin sont des littéraux d'horodatage ionique qui peuvent être indiqués par des crochets inverses (`` `...` ``). Pour en savoir plus, consultez Interroger Ion avec PartiQL dans Amazon QLDB.

Ces paramètres temporels adoptent le comportement suivant :

- L'heure de début et l'heure de fin sont toutes deux inclusives. Ils doivent être au format date et heure ISO 8601 et au format temps universel coordonné (UTC).
- L'heure de début doit être inférieure ou égale à l'heure de fin et peut être une date quelconque dans le passé.

- ID ID ID ID ID ID. ID ID.
- Si vous spécifiez une heure de début mais pas une heure de fin, votre requête définit par défaut la date et l'heure actuelles comme date et heure de fin. Si vous ne spécifiez aucune de ces options, votre requête renvoie l'historique complet.

« *identifiant* »

(Facultatif) L'ID du document pour lequel vous souhaitez interroger l'historique des révisions, indiqué par des guillemets simples.

> **Tip**
>
> Il est recommandé de qualifier une requête d'historique à la fois avec une plage de dates (heure de début et heure de fin) et un identifiant de document (`metadata.id`). Dans QLDB, chaque`SELECT` requête est traitée dans le cadre d'une transaction et est soumise à un délai d'expiration de transaction.
> Les requêtes d'historique n'utilisent pas les index que vous créez dans une table. L'historique QLDB est indexé uniquement par identifiant de document, et vous ne pouvez pas créer d'index d'historique supplémentaires pour le moment. Les requêtes d'historique qui incluent une heure de début et une heure de fin bénéficient d'une qualification par plage de dates.

Exemple de requête d'historique

Pour consulter l'historique du document d'immatriculation du véhicule, utilisez celui`id` que vous avez précédemment enregistréMise à jour et suppression de documents. Par exemple, la requête d'historique suivante renvoie toutes les révisions de l'ID du document`ADR2L11fGsU4Jr4EqTdnQF` qui étaient actives entre`2019-06-05T00:00:00Z` et`2019-06-05T23:59:59Z`.

> **Note**
>
> N'oubliez pas que les paramètres d'heure de début et de fin ne précisent pas la plage horaire au cours de laquelle les révisions ont été validées dans le journal lors d'une transaction. Par exemple, si une révision a été validée auparavant`2019-06-05T00:00:00Z` et est restée active après cette heure de début, cet exemple de requête renverra cette révision dans les résultats.

Assurez-vous de remplacer l'`id`heure de début et l'heure de fin par vos propres valeurs, le cas échéant.

```
SELECT * FROM history(VehicleRegistration, `2019-06-05T00:00:00Z`,
 `2019-06-05T23:59:59Z`) AS h
WHERE h.metadata.id = 'ADR2Ll1fGsU4Jr4EqTdnQF' --replace with your id
```

Les résultats de votre requête doivent se présenter comme suit :

```
{
    blockAddress:{
        strandId:"JdxjkR9bSYB5jMHWcI464T",
        sequenceNo:14
    },
    hash:{{B2wYwrHKOWsmIBmxUgPRrTx9lv36tMlod2xVvWNiTbo=}},
    data: {
        VIN: "1HVBBAANXWH544237",
        LicensePlateNumber: "LS477D",
        State: "WA",
        City: "Tacoma",
        PendingPenaltyTicketAmount: 42.20,
        ValidFromDate: 2011-10-26T,
        ValidToDate: 2023-09-25T,
        Owners: {
            PrimaryOwner: { PersonId: "KmA3XPKKFqYCP2zhR3d0Ho" },
            SecondaryOwners: []
        }
    },
    metadata:{
        id:"ADR2Ll1fGsU4Jr4EqTdnQF",
        version:0,
        txTime:2019-06-05T20:53:321d-3Z,
        txId:"HgXAkLjAtV0HQ4lNYdzX60"
    }
},
{
    blockAddress:{
        strandId:"JdxjkR9bSYB5jMHWcI464T",
        sequenceNo:17
    },
    hash:{{LGSFZ4iEYWZeMwmAqcxxNyT4wbCtuMOmFCj8pEd6Mp0=}},
    data: {
        VIN: "1HVBBAANXWH544237",
```

```
        LicensePlateNumber: "LS477D",
        State: "WA",
        PendingPenaltyTicketAmount: 42.20,
        ValidFromDate: 2011-10-26T,
        ValidToDate: 2023-09-25T,
        Owners: {
            PrimaryOwner: { PersonId: "KmA3XPKKFqYCP2zhR3d0Ho" },
            SecondaryOwners: []
        },
        City: "Bellevue"
    },
    metadata:{
        id:"ADR2Ll1fGsU4Jr4EqTdnQF",
        version:1,
        txTime:2019-06-05T21:01:442d-3Z,
        txId:"9cArhIQV5xf5Tf5vtsPwPq"
    }
},
{
    blockAddress:{
        strandId:"JdxjkR9bSYB5jMHWcI464T",
        sequenceNo:19
    },
    hash:{{7bm5DUwpqJFGrmZpb7h9wAxtvggYLPcXq+LAobi9fDg=}},
    metadata:{
        id:"ADR2Ll1fGsU4Jr4EqTdnQF",
        version:2,
        txTime:2019-06-05T21:03:76d-3Z,
        txId:"9GslbtDtpVHAgYghR5FXbZ"
    }
}
```

La sortie inclut des attributs de métadonnées qui fournissent des détails sur le moment où chaque élément a été modifié et par quelle transaction. À partir de ces données, vous pouvez voir ce qui suit :

- Le document est identifié de manière unique par son attribut système`id` :`ADR2Ll1fGsU4Jr4EqTdnQF`. Il s'agit d'un UUID représenté dans une chaîne codée en Base62.
- Une`INSERT` instruction crée la révision initiale d'un document (version`0`).
- Chaque mise à jour ultérieure crée une nouvelle révision avec le même document`id` et un numéro de version incrémenté.

- `LetxId` champ indique la transaction qui a validé chaque révision et`txTime` indique quand chaque révision a été validée.
- Une`DELETE` instruction crée une nouvelle révision, mais définitive, d'un document. Cette révision finale ne contient que des métadonnées.

Pour savoir comment supprimer définitivement une révision, passez àRédaction de révisions de documents.

Rédaction de révisions de documents

Dans Amazon QLDB, une`DELETE` instruction ne supprime logiquement un document qu'en créant une nouvelle révision qui le marque comme supprimé. QLDB prend également en charge une opération de rédaction de données qui vous permet de supprimer définitivement les révisions de documents inactives dans l'historique d'un tableau.

> **Note**
>
> Tous les registres créés avant le 22 juillet 2021 ne sont actuellement pas éligibles à la rédaction. Vous pouvez consulter l'heure de création de votre registre sur la console Amazon QLDB.

L'opération de rédaction supprime uniquement les données utilisateur de la révision spécifiée et laisse la séquence du journal et les métadonnées du document inchangées. Cela permet de préserver l'intégrité globale des données de votre registre.

Avant de commencer à rédiger des données dans QLDB, assurez-vous de consulterConsidérations et restrictions relatives à la rédaction et restrictions relatives la référence Amazon QLDB PartiQL.

Rubriques

- Procédures stockées de rédaction
- Vérifier si une rédaction est complète
- Exemple de rédaction
- Supprimer et rédiger une révision active
- Supprimer un champ particulier dans une révision

Procédures stockées de rédaction

Vous pouvez utiliser la procédureREDACT_REVISION enregistrée pour supprimer définitivement une révision individuelle et inactive dans un registre. Cette procédure stockée supprime toutes les données utilisateur de la révision spécifiée à la fois dans le stockage indexé et dans le stockage journal. Toutefois, la séquence du journal et les métadonnées du document, y compris l'identifiant et le hachage du document, restent inchangées. Cette opération est irréversible.

La révision du document spécifiée doit être une révision inactive dans l'historique. La dernière révision active d'un document n'est pas éligible à la rédaction.

Pour supprimer plusieurs révisions, vous devez exécuter la procédure enregistrée une fois pour chaque révision. Vous pouvez supprimer une révision par transaction.

Syntaxe

```
EXEC REDACT_REVISION `block-address`, 'table-id', 'document-id'
```

Arguments

`*adresse de bloc*`

Emplacement du bloc de journal de la révision du document à rédiger. Une adresse est une structure Amazon Ion qui comporte deux champs :`strandId` et`sequenceNo`.

Il s'agit d'une valeur littérale d'ions qui est indiquée par des coches inverses. Par exemple :

```
`{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`
```

« *ID de table* »

L'ID unique du tableau dont vous souhaitez modifier la version du document, indiqué par des guillemets simples.

« *identifiant du document* »

L'ID de document unique de la révision à rédiger, indiqué par des guillemets simples.

Vérifier si une rédaction est complète

Lorsque vous soumettez une demande de rédaction en exécutant la procédure stockée, QLDB traite la rédaction des données de manière asynchrone. Une fois la révision terminée, les données utilisateur (représentées par la`data` structure) sont supprimées définitivement. Pour vérifier si une demande de rédaction est terminée, vous pouvez utiliser l'une des méthodes suivantes :

- Exportation de journaux
- Flux de journaux
- GetBlock Opération d'API
- GetRevision Opération d'API
- Fonction historique— Remarque : Une fois qu'une rédaction est terminée dans le journal, cela peut prendre un certain temps avant que les requêtes d'historique n'affichent le résultat de la rédaction. Il se peut que certaines révisions soient expurgées avant d'autres lorsque la rédaction asynchrone est terminée, mais les requêtes d'historique finiront par afficher les résultats complets.

Une fois la rédaction d'une révision terminée, la`data` structure de la révision est remplacée par un nouveau`dataHash` champ. La valeur de ce champ est le hachage ionique de la`data` structure supprimée, comme illustré dans l'exemple suivant. Par conséquent, le registre conserve l'intégrité globale de ses données et reste vérifiable cryptographiquement par le biais des opérations d'API de vérification existantes. Pour en savoir plus sur la validation, consultezVérification des données dans Amazon QLDB.

Exemple de rédaction

Tenez compte du document d'immatriculation du véhicule que vous avez consulté précédemmentConsultation de l'historique des révisions. Supposons que vous souhaitiez biffer la deuxième révision (`version:1`). L'exemple de requête suivant montre cette révision avant la rédaction. Dans les résultats de la requête, la`data` structure qui sera supprimée est surlignée en *`italique rouge`*.

```
SELECT * FROM history(VehicleRegistration) AS h
WHERE h.metadata.id = 'ADR2Ll1fGsU4Jr4EqTdnQF' --replace with your id
AND h.metadata.version = 1
```

```
{
```

```
    blockAddress:{
        strandId:"JdxjkR9bSYB5jMHWcI464T",
        sequenceNo:17
    },
    hash:{{LGSFZ4iEYWZeMwmAqcxxNyT4wbCtuMOmFCj8pEd6Mp0=}},
    data: {
        VIN: "1HVBBAANXWH544237",
        LicensePlateNumber: "LS477D",
        State: "WA",
        PendingPenaltyTicketAmount: 42.20,
        ValidFromDate: 2011-10-26T,
        ValidToDate: 2023-09-25T,
        Owners: {
            PrimaryOwner: { PersonId: "KmA3XPKKFqYCP2zhR3d0Ho" },
            SecondaryOwners: []
        },
        City: "Bellevue"
    },
    metadata:{
        id:"ADR2Ll1fGsU4Jr4EqTdnQF",
        version:1,
        txTime:2019-06-05T21:01:442d-3Z,
        txId:"9cArhIQV5xf5Tf5vtsPwPq"
    }
}
```

Notez le`blockAddress` dans les résultats de la requête, car vous devez transmettre cette valeur à la procédure`REDACT_REVISION` stockée. Recherchez ensuite l'identifiant unique de la`VehicleRegistration` table en interrogeant le catalogue du système, comme suit.

```
SELECT tableId FROM information_schema.user_tables
WHERE name = 'VehicleRegistration'
```

Utilisez cet ID de table ainsi que l'ID du document et l'adresse du bloc pour exécuter`REDACT_REVISION`. L'identifiant de la table et l'identifiant du document sont des chaînes littérales qui doivent être placées entre guillemets simples, et l'adresse du bloc est un littéral Ion entouré de crochets inverses. Veillez à remplacer ces arguments par vos propres valeurs, le cas échéant.

```
EXEC REDACT_REVISION `{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`,
 '5PLf9SXwndd63lPaSIa0O6', 'ADR2Ll1fGsU4Jr4EqTdnQF'
```

> **Tip**
>
> Lorsque vous utilisez la console QLDB ou le shell QLDB pour demander un identifiant de table ou un identifiant de document (ou toute valeur littérale de chaîne), la valeur renvoyée est placée entre guillemets doubles. Toutefois, lorsque vous spécifiez les arguments d'ID de table et d'ID de document de la procédure`REDACT_REVISION` stockée, vous devez placer les valeurs entre guillemets simples.
>
> Cela est dû au fait que vous écrivez des instructions au format PartiQL, mais QLDB renvoie les résultats au format Amazon Ion. Pour plus de détails sur la syntaxe et la sémantique de PartiQL dans QLDB, consultezInterroger Ion avec PartiQL.

Une demande de rédaction valide renvoie une structure ionique qui représente la révision du document que vous êtes en train de biffer, comme suit.

```
{
  blockAddress: {
    strandId: "JdxjkR9bSYB5jMHWcI464T",
    sequenceNo: 17
  },
  tableId: "5PLf9SXwndd63lPaSIa0O6",
  documentId: "ADR2Ll1fGsU4Jr4EqTdnQF",
  version: 1
}
```

Lorsque vous exécutez cette procédure stockée, QLDB traite votre demande de rédaction de manière asynchrone. Une fois la rédaction terminée, la`data` structure est définitivement supprimée et remplacée par un nouveau *`dataHash`*champ. La valeur de ce champ est le hachage ionique de la`data` structure supprimée, comme suit.

> **Note**
>
> Cet`dataHash` exemple est fourni à titre informatif uniquement et ne constitue pas une véritable valeur de hachage calculée.

```
{
    blockAddress:{
        strandId:"JdxjkR9bSYB5jMHWcI464T",
```

```
        sequenceNo:17
    },
    hash:{{LGSFZ4iEYWZeMwmAqcxxNyT4wbCtuMOmFCj8pEd6Mp0=}},
    dataHash: {{s83jd7sfhsdfhksj7hskjdfjfpIPP/DP2hvionas2d4=}},
    metadata:{
        id:"ADR2Ll1fGsU4Jr4EqTdnQF",
        version:1,
        txTime:2019-06-05T21:01:442d-3Z,
        txId:"9cArhIQV5xf5Tf5vtsPwPq"
    }
}
```

Supprimer et rédiger une révision active

Les révisions actives des documents (c'est-à-dire les dernières révisions non supprimées de chaque document) ne sont pas éligibles à la rédaction des données. Avant de pouvoir rédiger une révision active, vous devez d'abord la mettre à jour ou la supprimer. Cela déplace la révision précédemment active vers l'historique et la rend éligible à la rédaction.

Si votre cas d'utilisation nécessite que l'ensemble du document soit marqué comme supprimé, vous devez d'abord utiliser une instruction DELETE. Par exemple, l'instruction suivante supprime logiquement le`VehicleRegistration` document dont le VIN est`1HVBBAANXWH544237`.

```
DELETE FROM VehicleRegistration AS r
WHERE r.VIN = '1HVBBAANXWH544237'
```

Ensuite, rédigez la révision précédente avant cette suppression, comme décrit précédemment. Si nécessaire, vous pouvez également modifier individuellement les révisions précédentes.

Si votre cas d'utilisation nécessite que le document reste actif, vous devez d'abord utiliser une instruction UPDATE ou FROM pour masquer ou supprimer les champs que vous souhaitez biffer. Ce processus est décrit dans la section suivante.

Supprimer un champ particulier dans une révision

QLDB ne prend pas en charge la rédaction d'un champ particulier dans une révision de document. Pour ce faire, vous pouvez d'abord utiliser une instruction UPDATE-REMOVE ou FROM-REMOVE pour supprimer un champ existant d'une révision. Par exemple, l'instruction suivante supprime le`LicensePlateNumber` champ du`VehicleRegistration` document dont le VIN est`1HVBBAANXWH544237`.

```
UPDATE VehicleRegistration AS r
REMOVE r.LicensePlateNumber
WHERE r.VIN = '1HVBBAANXWH544237'
```

Ensuite, rédigez la révision précédente avant cette suppression, comme décrit précédemment. Si nécessaire, vous pouvez également supprimer individuellement toutes les révisions précédentes qui incluent ce champ désormais supprimé.

Pour savoir comment optimiser vos requêtes, passez àOptimisation des performances des données.

Optimisation des performances des données

Amazon QLDB est destiné à répondre aux besoins de charges de travail de traitement transactionnel en ligne (OLTP) à hautes performances. Cela signifie que QLDB est optimisé pour un ensemble spécifique de modèles de requêtes, même s'il prend en charge des fonctionnalités de requête de type SQL. Il est essentiel de concevoir des applications et leurs modèles de données pour qu'ils fonctionnent avec ces modèles de requêtes. Sinon, au fur et à mesure que vos tables s'agrandissent, vous rencontrerez d'importants problèmes de performances, notamment la latence des requêtes, les délais d'expiration des transactions et les conflits de simultanéité.

Cette section décrit les contraintes de requête dans QLDB et fournit des conseils pour écrire des requêtes optimales compte tenu de ces contraintes.

Rubriques

- Délai d'expiration des transactions
- Conflits de concurrence
- Modèles de données optimaux
- Modèles de données à éviter
- Surveillance des performances

Délai d'expiration des transactions

Dans QLDB, chaque instruction partiQL (y compris chaque`SELECT` requête) est traitée dans le cadre d'une transaction et est soumise à un délai d'expiration de transaction. Une transaction peut s'exécuter pendant 30 secondes au maximum avant d'être validée. Au-delà de cette limite, QLDB rejette tout travail effectué sur la transaction et supprime la session qui a exécuté la transaction. Cette

limite protège le client du service contre les fuites de sessions en démarrant des transactions sans les valider ni les annuler.

Conflits de concurrence

QLDB implémente le contrôle de la concurrence en utilisant un contrôle de concurrence optimiste (OCC). Les requêtes sous-optimales peuvent également entraîner davantage de conflits OCC. Pour plus d'informations sur l'OCC, consultezModèle de mise QLDB simultansimultansimultansimultansimultansimultansimultansimultansimultanéité.

Modèles de données optimaux

Il est recommandé d'exécuter des instructions avec une clause de`WHERE` prédicat qui filtre en fonction d'un champ indexé ou d'un identifiant de document. QLDB nécessite un opérateur d'égalité (`=`ou`IN`) sur un champ indexé pour rechercher efficacement un document.

Vous trouverez ci-dessous des exemples de modèles de requêtes optimaux dans la vue utilisateur.

```
--Indexed field (VIN) lookup using the = operator
SELECT * FROM VehicleRegistration
WHERE VIN = '1N4AL11D75C109151'

--Indexed field (VIN) AND non-indexed field (City) lookup
SELECT * FROM VehicleRegistration
WHERE VIN = '1N4AL11D75C109151' AND City = 'Seattle'

--Indexed field (VIN) lookup using the IN operator
SELECT * FROM VehicleRegistration
WHERE VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')

--Document ID (r_id) lookup using the BY clause
SELECT * FROM VehicleRegistration BY r_id
WHERE r_id = '3Qv67yjXEwB9SjmvkuG6Cp'
```

Toute requête qui ne suit pas ces modèles appelle une analyse complète de la table. Les analyses de tables peuvent entraîner l'expiration des délais de transaction pour les requêtes portant sur des tables volumineuses ou les requêtes renvoyant de grands ensembles de résultats. Ils peuvent également entraîner des conflits OCC avec des transactions concurrentes.

Indices de cardinalité élevée

Nous recommandons d'indexer les champs contenant des valeurs de cardinalité élevées. Par exemple, les`LicensePlateNumber` champs`VIN` et de la`VehicleRegistration` table sont des champs indexés destinés à être uniques.

Évitez d'indexer les champs à faible cardinalité tels que les codes de statut, les adresses, les États ou les provinces et les codes postaux. Si vous indexez un tel champ, vos requêtes peuvent produire des ensembles de résultats volumineux qui sont plus susceptibles d'entraîner l'expiration des délais de transaction ou de provoquer des conflits OCC involontaires.

Requêtes de vue validées

Les requêtes que vous exécutez dans la vue validée suivent les mêmes directives d'optimisation que les requêtes de vue utilisateur. Les index que vous créez dans une table sont également utilisés pour les requêtes dans la vue validée.

Requêtes relatives aux fonctions d'historique

Les requêtes des fonctions d'historique n'utilisent pas les index que vous créez dans une table. L'historique QLDB est indexé uniquement par identifiant de document, et vous ne pouvez pas créer d'index d'historique supplémentaires pour le moment.

Il est recommandé de qualifier une requête d'historique à la fois avec une plage de dates (heure de début et heure de fin) et un identifiant de document (`metadata.id`). Les requêtes d'historique qui incluent une heure de début et une heure de fin bénéficient d'une qualification par plage de dates.

Requêtes de jointure internes

Pour les requêtes de jointure internes, utilisez des critères de jointure qui incluent au moins un champ indexé pour la table située sur le côté droit de la jointure. Sans index de jointure, une requête de jointure appelle plusieurs analyses de table : pour chaque document de la table de gauche de la jointure, la requête analyse entièrement la table de droite. La meilleure pratique consiste à effectuer une jointure sur des champs indexés pour chaque table que vous joignez, en plus de spécifier un prédicat d'`WHERE`égalité pour au moins une table.

Par exemple, la requête suivante joint les`Vehicle` tables`VehicleRegistration` et de leurs`VIN` champs respectifs, qui sont tous deux indexés. Cette requête comporte également un prédicat d'égalité sur`VehicleRegistration.VIN`.

```
SELECT * FROM VehicleRegistration AS r INNER JOIN Vehicle AS v
ON r.VIN = v.VIN
```

```
WHERE r.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

Choisissez des indices de cardinalité élevés pour les critères de jointure et les prédicats d'égalité dans vos requêtes de jointure.

Modèles de données à éviter

Vous trouverez ci-dessous quelques exemples d'instructions sous-optimales qui ne sont pas adaptées aux tables de grande taille dans QLDB. Nous vous recommandons vivement de ne pas vous fier à ce type de requêtes pour les tables qui s'agrandissent au fil du temps, car vos requêtes finiront par entraîner des délais d'expiration des transactions. Comme les tableaux contiennent des documents dont la taille varie, il est difficile de définir des limites précises pour les requêtes non indexées.

```
--No predicate clause
SELECT * FROM Vehicle

--COUNT() is not an optimized function
SELECT COUNT(*) FROM Vehicle

--Low-cardinality predicate
SELECT * FROM Vehicle WHERE Color = 'Silver'

--Inequality (>) does not qualify for indexed lookup
SELECT * FROM Vehicle WHERE "Year" > 2019

--Inequality (LIKE)
SELECT * FROM Vehicle WHERE VIN LIKE '1N4AL%'

--Inequality (BETWEEN)
SELECT SUM(PendingPenaltyTicketAmount) FROM VehicleRegistration
WHERE ValidToDate BETWEEN `2020-01-01T` AND `2020-07-01T`

--No predicate clause
DELETE FROM Vehicle

--No document id, and no date range for the history() function
SELECT * FROM history(Vehicle)
```

En général, nous ne recommandons pas d'exécuter les types de modèles de requête suivants pour les cas d'utilisation en production dans QLDB :

- Requêtes de traitement analytique en ligne (OLAP)
- Requêtes exploratoires sans clause de prédicat
- Requêtes de signalement
- Recherche de texte

Nous vous recommandons plutôt de diffuser vos données vers un service de base de données spécialement conçu et optimisé pour les cas d'utilisation analytiques. Par exemple, vous pouvez diffuser des données QLDB vers Amazon OpenSearch Service pour fournir des fonctionnalités de recherche en texte intégral sur les documents. Pour un exemple d'application illustrant ce cas d'utilisation, consultez le GitHub référentiel aws-samples/amazon-qldb-streaming-amazon - opensearch-service-sample-python. Pour plus d'informations sur les flux QLDB, consultezDiffusion en continu de données de journaux depuis Amazon QLDB.

Surveillance des performances

Le pilote QLDB fournit des informations de synchronisation et d'utilisation des E/S dans l'objet de résultat d'une instruction. Vous pouvez utiliser ces métriques pour identifier les instructions PartiQL inefficaces. Pour en savoir plus, passez àObtention des statistiques d'instruction PartiQL.

Vous pouvez également utiliser Amazon CloudWatch pour suivre les performances de votre registre pour les opérations de données. Surveillez la`CommandLatency` métrique pour un`LedgerName` et`CommandType`. Pour plus d'informations, veuillez consulter Surveillance avec Amazon CloudWatch. Pour savoir comment QLDB utilise les commandes pour gérer les opérations de données, consultezGestion des sessions avec le chauffeur.

Obtention des statistiques d'instruction PartiQL

Amazon QLDB fournit des statistiques d'exécution des instructions qui peuvent vous aider à optimiser votre utilisation de QLDB en exécutant des instructions partiQL plus efficaces. QLDB renvoie ces statistiques ainsi que les résultats de l'instruction. Ils incluent des mesures qui quantifient l'utilisation des E/S consommées et le temps de traitement côté serveur, que vous pouvez utiliser pour identifier les déclarations inefficaces.

Cette fonctionnalité est actuellement disponible dans l'éditeur PartiQL de la console QLDB, dans le shell QLDB et dans la dernière version du pilote QLDB pour toutes les langues prises en charge. Vous pouvez également consulter les statistiques des relevés pour l'historique de vos requêtes sur la console.

Rubriques

Utilisation des E/S

La métrique d'utilisation des E/S décrit le nombre de demandes d'E/S de lecture. Si le nombre de demandes d'E/S de lecture est plus élevé que prévu, cela indique que l'instruction n'est pas optimisée, par exemple en l'absence d'index. Nous vous recommandons de consulterModèles de données optimaux la rubrique précédente intitulée Optimisation des performances des requêtes.

> **Note**
>
> Lorsque vous exécutez une`CREATE INDEX` instruction sur une table non vide, la métrique d'utilisation des E/S inclut uniquement les demandes de lecture pour l'appel de création d'index synchrone.
> QLDB construit l'index de tous les documents existants dans la table de manière asynchrone. Ces demandes de lecture asynchrones ne sont pas incluses dans la métrique d'utilisation des E/S à partir des résultats de vos relevés. Les demandes de lecture asynchrones sont facturées séparément et sont ajoutées au total de vos E/S de lecture une fois la construction de l'index terminée.

Utilisation de la console QLDB

Pour connaître l'utilisation des E/S en lecture d'une instruction à l'aide de la console QLDB, procédez comme suit :

1. Ouvrez la console Amazon QLDB à l'adresse https://console.aws.amazon.com/qldb.
2. Dans le panneau de navigation, choisissez PartiQL editor.
3. Choisissez un livre dans la liste déroulante des registres.
4. Dans la fenêtre de l'éditeur de requêtes, entrez l'instruction de votre choix, puis choisissez Exécuter. Voici un exemple de requête.

```
SELECT * FROM testTable WHERE firstName = 'Jim'
```

Pour exécuter une instruction, vous pouvez également utiliser le raccourci clavierCtrl +Enter pour Windows ouCmd +Return pour macOS. Pour plus de raccourcis clavier, consultezRaccourcis clavier de l'éditeur PartiQL.

5. Sous la fenêtre de l'éditeur de requêtes, les résultats de votre requête incluent les E/S de lecture, c'est-à-dire le nombre de demandes de lecture effectuées par l'instruction.

Vous pouvez également consulter les E/S de lecture de l'historique de vos requêtes en procédant comme suit :

1. Dans le volet de navigation, choisissez Requêtes récentes sous l'éditeur PartiQL.
2. La colonne Read I/Os affiche le nombre de demandes de lecture effectuées par chaque instruction.

Utilisation du pilote QLDB

Pour connaître l'utilisation des E/S d'une instruction à l'aide du pilote QLDB, appelez l'`getConsumedIO`opération du curseur de flux ou du curseur mis en mémoire tampon du résultat.

Les exemples de code suivants montrent comment obtenir des E/S lues à partir du curseur de flux d'un résultat d'instruction.

Java

```
import com.amazon.ion.IonSystem;
import com.amazon.ion.IonValue;
import com.amazon.ion.system.IonSystemBuilder;
import software.amazon.qldb.IOUsage;
import software.amazon.qldb.Result;

IonSystem ionSystem = IonSystemBuilder.standard().build();
IonValue ionFirstName = ionSystem.newString("Jim");

driver.execute(txn -> {
    Result result = txn.execute("SELECT * FROM testTable WHERE firstName = ?",
 ionFirstName);

    for (IonValue ionValue : result) {
        // User code here to handle results
    }
```

```
    IOUsage ioUsage = result.getConsumedIOs();
    long readIOs = ioUsage.getReadIOs();
});
```

.NET

```
using Amazon.IonDotnet.Builders;
using Amazon.IonDotnet.Tree;
using Amazon.QLDB.Driver;
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;

// This is one way of creating Ion values. We can also use a ValueFactory.
// For more details, see: https://docs.aws.amazon.com/qldb/latest/developerguide/
driver-cookbook-dotnet.html#cookbook-dotnet.ion
IIonValue ionFirstName = IonLoader.Default.Load("Jim");

await driver.Execute(async txn =>
{
    IAsyncResult result = await txn.Execute("SELECT * FROM testTable WHERE firstName
 = ?", ionFirstName);

    // Iterate through stream cursor to accumulate read IOs.
    await foreach (IIonValue ionValue in result)
    {
        // User code here to handle results.
        // Warning: It is bad practice to rely on results within a lambda block,
 unless
        // it is to check the state of a result. This is because lambdas are
 retryable.
    }

    var ioUsage = result.GetConsumedIOs();
    var readIOs = ioUsage?.ReadIOs;
});
```

> **Note**
>
> Pour convertir en code synchrone, supprimez les`async` mots-clés`await` et, puis remplacez le`IAsyncResult` type par`IResult`.

Go

```
import (
    "context"
    "fmt"
    "github.com/awslabs/amazon-qldb-driver-go/v2/qldbdriver"
)

driver.Execute(context.Background(), func(txn qldbdriver.Transaction) (interface{},
 error) {
    result, err := txn.Execute("SELECT * FROM testTable WHERE firstName = ?", "Jim")

    if err != nil {
        panic(err)
    }

    for result.Next(txn) {
        // User code here to handle results
    }

    ioUsage := result.GetConsumedIOs()
    readIOs := *ioUsage.GetReadIOs()
    fmt.Println(readIOs)
    return nil,nil
})
```

Node.js

```
import { IOUsage, ResultReadable, TransactionExecutor } from "amazon-qldb-driver-
nodejs";

await driver.executeLambda(async (txn: TransactionExecutor) => {
    const result: ResultReadable = await txn.executeAndStreamResults("SELECT * FROM
 testTable WHERE firstName = ?", "Jim");

    for await (const chunk of result) {
        // User code here to handle results
    }

    const ioUsage: IOUsage = result.getConsumedIOs();
    const readIOs: number = ioUsage.getReadIOs();
});
```

Python

```
def get_read_ios(transaction_executor):
    cursor = transaction_executor.execute_statement("SELECT * FROM testTable WHERE
 firstName = ?", "Jim")

    for row in cursor:
        # User code here to handle results
        pass

    consumed_ios = cursor.get_consumed_ios()
    read_ios = consumed_ios.get('ReadIOs')

qldb_driver.execute_lambda(lambda txn: get_read_ios(txn))
```

Les exemples de code suivants montrent comment obtenir des E/S lues à partir du curseur en mémoire tampon du résultat d'une instruction. Cela renvoie le nombre total d'E/S de lectureExecuteStatement et deFetchPage demandes.

Java

```
import com.amazon.ion.IonSystem;
import com.amazon.ion.IonValue;
import com.amazon.ion.system.IonSystemBuilder;
import software.amazon.qldb.IOUsage;
import software.amazon.qldb.Result;

IonSystem ionSystem = IonSystemBuilder.standard().build();
IonValue ionFirstName = ionSystem.newString("Jim");

Result result = driver.execute(txn -> {
    return txn.execute("SELECT * FROM testTable WHERE firstName = ?", ionFirstName);
});

IOUsage ioUsage = result.getConsumedIOs();
long readIOs = ioUsage.getReadIOs();
```

.NET

```
using Amazon.IonDotnet.Builders;
using Amazon.IonDotnet.Tree;
```

```
using Amazon.QLDB.Driver;
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;

IIonValue ionFirstName = IonLoader.Default.Load("Jim");

IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("SELECT * FROM testTable WHERE firstName = ?",
 ionFirstName);
});

var ioUsage = result.GetConsumedIOs();
var readIOs = ioUsage?.ReadIOs;
```

> **Note**
>
> Pour convertir en code synchrone, supprimez les`async` mots-clés`await` et, puis remplacez le`IAsyncResult` type par`IResult`.

Go

```
import (
    "context"
    "fmt"
    "github.com/awslabs/amazon-qldb-driver-go/v2/qldbdriver"
)

result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
    result, err :=  txn.Execute("SELECT * FROM testTable WHERE firstName = ?",
 "Jim")
    if err != nil {
        return nil, err
    }
    return txn.BufferResult(result)
})

if err != nil {
    panic(err)
}
```

```
qldbResult := result.(*qldbdriver.BufferedResult)
ioUsage := qldbResult.GetConsumedIOs()
readIOs := *ioUsage.GetReadIOs()
fmt.Println(readIOs)
```

Node.js

```
import { IOUsage, Result, TransactionExecutor } from "amazon-qldb-driver-nodejs";

const result: Result = await driver.executeLambda(async (txn: TransactionExecutor)
 => {
    return await txn.execute("SELECT * FROM testTable WHERE firstName = ?", "Jim");
});

const ioUsage: IOUsage = result.getConsumedIOs();
const readIOs: number = ioUsage.getReadIOs();
```

Python

```
cursor = qldb_driver.execute_lambda(
    lambda txn: txn.execute_statement("SELECT * FROM testTable WHERE firstName = ?",
 "Jim"))

consumed_ios = cursor.get_consumed_ios()
read_ios = consumed_ios.get('ReadIOs')
```

> **Note**
>
> Le curseur de flux est dynamique car il pagine le jeu de résultats. Par conséquent, les`getTimingInformation` opérations`getConsumedIOs` et renvoient les mesures accumulées à partir du moment où vous les appelez.
> Le curseur mis en mémoire tampon met en mémoire le jeu de résultats et renvoie le total des mesures cumulées.

Informations de chronométrage

La métrique d'informations de synchronisation décrit le temps de traitement côté serveur en millisecondes. Le temps de traitement côté serveur est défini comme le temps que QLDB consacre

au traitement d'une instruction. Cela n'inclut pas le temps passé sur les appels réseau ni les pauses. Cette métrique élimine l'ambiguïté entre le temps de traitement côté service QLDB et le temps de traitement côté client.

Utilisation de la console QLDB

Pour obtenir les informations de synchronisation d'une instruction à l'aide de la console QLDB, procédez comme suit :

1. Ouvrez la console Amazon QLDB à l'adresse https://console.aws.amazon.com/qldb.
2. Dans le panneau de navigation, choisissez PartiQL editor.
3. Choisissez un livre dans la liste déroulante des registres.
4. Dans la fenêtre de l'éditeur de requêtes, entrez l'instruction de votre choix, puis choisissez Exécuter. Voici un exemple de requête.

   ```
   SELECT * FROM testTable WHERE firstName = 'Jim'
   ```

   Pour exécuter une instruction, vous pouvez également utiliser le raccourci clavierCtrl +Enter pour Windows ouCmd +Return pour macOS. Pour plus de raccourcis clavier, consultezRaccourcis clavier de l'éditeur PartiQL.
5. Sous la fenêtre de l'éditeur de requêtes, les résultats de votre requête incluent la latence côté serveur, c'est-à-dire le délai entre la réception de la demande d'instruction par QLDB et l'envoi de la réponse. Il s'agit d'un sous-ensemble de la durée totale de la requête.

Vous pouvez également consulter les informations temporelles de l'historique de vos requêtes en procédant comme suit :

1. Dans le volet de navigation, choisissez Requêtes récentes sous l'éditeur PartiQL.
2. La colonne Temps d'exécution (ms) affiche ces informations de synchronisation pour chaque instruction.

Utilisation du pilote QLDB

Pour obtenir les informations de synchronisation d'une instruction à l'aide du pilote QLDB, appelez l'`getTimingInformation`opération du curseur de flux ou du curseur mis en mémoire tampon du résultat.

Les exemples de code suivants montrent comment obtenir le temps de traitement à partir du curseur de flux du résultat d'une instruction.

Java

```
import com.amazon.ion.IonSystem;
import com.amazon.ion.IonValue;
import com.amazon.ion.system.IonSystemBuilder;
import software.amazon.qldb.Result;
import software.amazon.qldb.TimingInformation;

IonSystem ionSystem = IonSystemBuilder.standard().build();
IonValue ionFirstName = ionSystem.newString("Jim");

driver.execute(txn -> {
    Result result = txn.execute("SELECT * FROM testTable WHERE firstName = ?",
 ionFirstName);

    for (IonValue ionValue : result) {
        // User code here to handle results
    }

    TimingInformation timingInformation = result.getTimingInformation();
    long processingTimeMilliseconds =
 timingInformation.getProcessingTimeMilliseconds();
});
```

.NET

```
using Amazon.IonDotnet.Builders;
using Amazon.IonDotnet.Tree;
using Amazon.QLDB.Driver;
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;

IIonValue ionFirstName = IonLoader.Default.Load("Jim");

await driver.Execute(async txn =>
{
    IAsyncResult result = await txn.Execute("SELECT * FROM testTable WHERE firstName
 = ?", ionFirstName);

    // Iterate through stream cursor to accumulate processing time.
    await foreach(IIonValue ionValue in result)
```

```
    {
        // User code here to handle results.
        // Warning: It is bad practice to rely on results within a lambda block,
 unless
        // it is to check the state of a result. This is because lambdas are
 retryable.
    }

    var timingInformation = result.GetTimingInformation();
    var processingTimeMilliseconds = timingInformation?.ProcessingTimeMilliseconds;
});
```

> **Note**
>
> Pour convertir en code synchrone, supprimez les`async` mots-clés`await` et, puis remplacez le`IAsyncResult` type par`IResult`.

Go

```
import (
    "context"
    "fmt"
    "github.com/awslabs/amazon-qldb-driver-go/v2/qldbdriver"
)

driver.Execute(context.Background(), func(txn qldbdriver.Transaction) (interface{},
 error) {
    result, err := txn.Execute("SELECT * FROM testTable WHERE firstName = ?", "Jim")

    if err != nil {
        panic(err)
    }

    for result.Next(txn) {
        // User code here to handle results
    }

    timingInformation := result.GetTimingInformation()
    processingTimeMilliseconds := *timingInformation.GetProcessingTimeMilliseconds()
    fmt.Println(processingTimeMilliseconds)
    return nil, nil
```

```
})
```

Node.js

```
import { ResultReadable, TimingInformation, TransactionExecutor } from "amazon-qldb-
driver-nodejs";

await driver.executeLambda(async (txn: TransactionExecutor) => {
    const result: ResultReadable = await txn.executeAndStreamResults("SELECT * FROM
 testTable WHERE firstName = ?", "Jim");

    for await (const chunk of result) {
        // User code here to handle results
    }

    const timingInformation: TimingInformation = result.getTimingInformation();
    const processingTimeMilliseconds: number =
 timingInformation.getProcessingTimeMilliseconds();
});
```

Python

```
def get_processing_time_milliseconds(transaction_executor):
    cursor = transaction_executor.execute_statement("SELECT * FROM testTable WHERE
 firstName = ?", "Jim")

    for row in cursor:
        # User code here to handle results
        pass

    timing_information = cursor.get_timing_information()
    processing_time_milliseconds =
 timing_information.get('ProcessingTimeMilliseconds')

qldb_driver.execute_lambda(lambda txn: get_processing_time_milliseconds(txn))
```

Les exemples de code suivants montrent comment obtenir le temps de traitement à partir du curseur en mémoire tampon du résultat d'une instruction. Cela renvoie le temps de traitement total des`FetchPage` demandes`ExecuteStatement` et des demandes.

Java

```
import com.amazon.ion.IonSystem;
import com.amazon.ion.IonValue;
import com.amazon.ion.system.IonSystemBuilder;
import software.amazon.qldb.Result;
import software.amazon.qldb.TimingInformation;

IonSystem ionSystem = IonSystemBuilder.standard().build();
IonValue ionFirstName = ionSystem.newString("Jim");

Result result = driver.execute(txn -> {
    return txn.execute("SELECT * FROM testTable WHERE firstName = ?", ionFirstName);
});

TimingInformation timingInformation = result.getTimingInformation();
long processingTimeMilliseconds = timingInformation.getProcessingTimeMilliseconds();
```

.NET

```
using Amazon.IonDotnet.Builders;
using Amazon.IonDotnet.Tree;
using Amazon.QLDB.Driver;
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;

IIonValue ionFirstName = IonLoader.Default.Load("Jim");

IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("SELECT * FROM testTable WHERE firstName = ?",
 ionFirstName);
});

var timingInformation = result.GetTimingInformation();
var processingTimeMilliseconds = timingInformation?.ProcessingTimeMilliseconds;
```

> **Note**
>
> Pour convertir en code synchrone, supprimez les`async` mots-clés`await` et, puis remplacez le`IAsyncResult` type par`IResult`.

Go

```
import (
    "context"
    "fmt"
    "github.com/awslabs/amazon-qldb-driver-go/v2/qldbdriver"
)

result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
    result, err :=  txn.Execute("SELECT * FROM testTable WHERE firstName = ?",
 "Jim")
    if err != nil {
        return nil, err
    }
    return txn.BufferResult(result)
})

if err != nil {
    panic(err)
}

qldbResult := result.(*qldbdriver.BufferedResult)
timingInformation := qldbResult.GetTimingInformation()
processingTimeMilliseconds := *timingInformation.GetProcessingTimeMilliseconds()
fmt.Println(processingTimeMilliseconds)
```

Node.js

```
import { Result, TimingInformation, TransactionExecutor } from "amazon-qldb-driver-
nodejs";

const result: Result = await driver.executeLambda(async (txn: TransactionExecutor)
 => {
    return await txn.execute("SELECT * FROM testTable WHERE firstName = ?", "Jim");
});

const timingInformation: TimingInformation = result.getTimingInformation();
const processingTimeMilliseconds: number =
 timingInformation.getProcessingTimeMilliseconds();
```

Python

```
cursor = qldb_driver.execute_lambda(
    lambda txn: txn.execute_statement("SELECT * FROM testTable WHERE firstName = ?",
 "Jim"))

timing_information = cursor.get_timing_information()
processing_time_milliseconds = timing_information.get('ProcessingTimeMilliseconds')
```

> **Note**
>
> Le curseur de flux est dynamique car il pagine le jeu de résultats. Par conséquent, les`getTimingInformation` opérations`getConsumedIOs` et renvoient les mesures accumulées à partir du moment où vous les appelez.
> Le curseur mis en mémoire tampon met en mémoire le jeu de résultats et renvoie le total des mesures cumulées.

Pour savoir comment interroger le catalogue système, passez àInterrogation du catalogue système.

Interrogation du catalogue système

Chaque table que vous créez dans un registre Amazon QLDB possède un identifiant unique attribué par le système. Vous pouvez trouver l'ID d'une table, sa liste d'index et d'autres métadonnées en interrogeant la table du catalogue système`information_schema.user_tables`.

Tous les identifiants attribués par le système sont des identifiants uniques universels (UUID) qui sont chacun représentés dans une chaîne codée en Base62. Pour plus d'informations, veuillez consulter Identifiants uniques dans Amazon QLDB.

L'exemple suivant montre les résultats d'une requête qui renvoie les attributs de métadonnées de la`VehicleRegistration` table.

```
SELECT * FROM information_schema.user_tables
WHERE name = 'VehicleRegistration'
```

```
{
    tableId: "5PLf9SXwndd63lPaSIa0O6",
```

```
    name: "VehicleRegistration",
    indexes: [
        { indexId: "Djg2nt0yIs2GY0T29Kud1z", expr: "[VIN]", status: "ONLINE" },
        { indexId: "4tPW3fUhaVhDinRgKRLhGU", expr: "[LicensePlateNumber]", status:
 "BUILDING" }
    ],
    status: "ACTIVE"
}
```

Champs de métadonnées de table

- `tableId`— ID unique de la table.
- `name`— Le nom de la table.
- `indexes`— La liste des index de la table.
  - `indexId`— ID unique de l'index.
  - `expr`— Le chemin du document indexé. Ce champ est une chaîne de la forme :`[fieldName]`.
  - `status`— État actuel de l'index (`BUILDING`FINALIZING,`ONLINE`,`FAILED`, ou`DELETING`). QLDB n'utilise pas l'index dans les requêtes tant que le statut ne l'est pas`ONLINE`.
  - `message`— Le message d'erreur qui décrit la raison pour laquelle l'index possède un`FAILED` statut. Ce champ est inclus uniquement pour les index défaillants.
- `status`— État actuel de la table (`ACTIVE`ou`INACTIVE`). Une table devient`INACTIVE` lorsque vous la`DROP` touchez.

Pour savoir comment gérer les tables à l'aide`UNDROP TABLE` des instructions`DROP TABLE` et, passez àGestion des tables.

Gestion des tables

Cette section explique comment gérer les tables à l'aide`UNDROP TABLE` des instructions`DROP TABLE` et d'Amazon QLDB. Il explique également comment étiqueter des tableaux pendant leur création. Les quotas relatifs au nombre de tables actives et au nombre total de tables que vous pouvez créer sont définis dansQuotas et limites d'Amazon QLDB.

Rubriques

- Consultation de l'historique des tables inactives
- Réactivation des tables

Baliser les tables lors de la création

> **Note**
>
> Le balisage des tables lors de leur création n'est actuellement pris en charge que pour les registres en mode`STANDARD` autorisations.

Vous pouvez baliser les ressources de table. Pour gérer les balises des tables existantes, utilisez les opérationsAWS Management Console ou de l'API`TagResource``UntagResource`, et`ListTagsForResource`. Pour plus d'informations, veuillez consulter Balisage des ressources Amazon QLDB.

Vous pouvez également définir des balises de table lors de la création de la table à l'aide de la console QLDB ou en les spécifiant dans une instruction`CREATE TABLE` PartiQL. L'exemple suivant crée une table nommée`Vehicle` avec la balise`environment=production`.

```
CREATE TABLE Vehicle WITH (aws_tags = `{'environment': 'production'}`)
```

En attribuant des étiquettes aux ressources au moment de la création, vous pouvez supprimer la nécessité d'exécuter des scripts d'identification personnalisés après la création de ressources. Une fois qu'un tableau a été balisé, vous pouvez contrôler l'accès au tableau à l'aide de ces balises. Par exemple, vous pouvez accorder un accès complet uniquement aux tables dotées d'une balise spécifique. Pour un exemple de politique JSON, consultezAccès complet à toutes les actions en fonction des balises du tableau.

Tables pliantes

Pour supprimer un tableau, utilisez uneDROP TABLE instruction de base. Lorsque vous déposez une table dans QLDB, vous ne faites que la désactiver.

Par exemple, l'instruction suivante désactive la`VehicleRegistration` table.

```
DROP TABLE VehicleRegistration
```

UneDROP TABLE instruction renvoie l'ID attribué par le système à la table. Le statut de`VehicleRegistration` doit désormais figurer`INACTIVE` dans la table du catalogue système information_schema.user_tables.

```
SELECT status FROM information_schema.user_tables
WHERE name = 'VehicleRegistration'
```

Consultation de l'historique des tables inactives

En plus d'un nom de table, vous pouvez également interroger le QLDBFonction historique avec un ID de table comme premier argument d'entrée. Vous devez utiliser l'ID de table pour consulter l'historique d'une table inactive. Une fois qu'une table est désactivée, vous ne pouvez plus interroger son historique à l'aide du nom de la table.

Commencez par rechercher l'ID de la table en interrogeant la table du catalogue système. Par exemple, la requête suivante renvoie la valeur`tableId` de la`VehicleRegistration` table.

```
SELECT tableId FROM information_schema.user_tables
WHERE name = 'VehicleRegistration'
```

Vous pouvez ensuite utiliser cet ID pour exécuter la même requête d'historique à partir deConsultation de l'historique des révisions. L'exemple suivant permet d'interroger l'historique de l'ID du document`ADR2Ll1fGsU4Jr4EqTdnQF` à partir de l'ID de table`5PLf9SXwndd63lPaSIa0O6`. L'ID de table est une chaîne littérale qui doit être placée entre guillemets simples.

```
--replace both the table and document IDs with your values
SELECT * FROM history('5PLf9SXwndd63lPaSIa0O6', `2000T`, `2019-06-05T23:59:59Z`) AS h
WHERE h.metadata.id = 'ADR2Ll1fGsU4Jr4EqTdnQF'
```

Réactivation des tables

Après avoir désactivé une table dans QLDB, vous pouvez utiliser l'TABLEAU DE DÉBALLAGEinstruction pour la réactiver.

Tout d'abord, recherchez l'identifiant de la table à partir de`information_schema.user_tables`. Par exemple, la requête suivante renvoie le`tableId` de la`VehicleRegistration` table. Le statut doit être`INACTIVE`.

```
SELECT tableId FROM information_schema.user_tables
```

```
WHERE name = 'VehicleRegistration'
```

Utilisez ensuite cet ID pour réactiver la table. Voici un exemple qui permet de supprimer l'ID de table`5PLf9SXwndd63lPaSIa006`. Dans ce cas, l'ID de table est un identifiant unique que vous placez entre guillemets doubles.

```
UNDROP TABLE "5PLf9SXwndd63lPaSIa006"
```

Le statut de`VehicleRegistration` devrait désormais être`ACTIVE`.

Pour savoir comment créer, décrire et supprimer des index, passez àGestion des index.

Gestion des index

Cette section décrit comment créer, décrire et supprimer des index dans Amazon QLDB. Le quota du nombre d'index que vous pouvez créer par table est défini dansQuotas et limites d'Amazon QLDB.

Rubriques

- Création d'index
- Description des index
- Supprimer des index
- Erreurs courantes

Création d'index

Comme décrit également dansCréation de tables et d'index, vous pouvez utiliser l'instruction CREATE INDEX pour créer un index dans une table pour un champ de niveau supérieur spécifié, comme suit. Le nom de la table et le nom du champ indexé distinguent les majuscules et minuscules.

```
CREATE INDEX ON VehicleRegistration (VIN)
```

```
CREATE INDEX ON VehicleRegistration (LicensePlateNumber)
```

Chaque index que vous créez sur une table possède un identifiant unique attribué par le système. Pour trouver cet ID d'index, consultez la section suivanteDescription des index.

> **Important**
>
> QLDB a besoin d'un index pour rechercher efficacement un document. Sans index, QLDB doit effectuer une analyse complète de la table lors de la lecture de documents. Cela peut entraîner des problèmes de performances sur des tables de grande taille, notamment des conflits de simultanéité et des délais de transaction.
> Pour éviter de scanner des tables, vous devez exécuter des instructions avec une clause de`WHERE` prédicat à l'aide d'un opérateur d'égalité (`=`ou`IN`) sur un champ indexé ou un identifiant de document. Pour plus d'informations, veuillez consulter Optimisation des performances des données.

Tenez compte des contraintes suivantes lors de la création d'index :

- Un index ne peut être créé que sur un seul champ de niveau supérieur. Les index composites, imbriqués, uniques et basés sur des fonctions ne sont pas pris en charge.
- Vous pouvez créer un index sur tous les types de données Ion, y compris`list` et`struct`. Toutefois, vous ne pouvez effectuer la recherche indexée que par égalité de la valeur totale des ions, quel que soit le type d'ion. Par exemple, lorsque vous utilisez un`list` type comme index, vous ne pouvez pas effectuer de recherche indexée par un élément de la liste.
- Les performances des requêtes ne sont améliorées que lorsque vous utilisez un prédicat d'égalité ; par exemple,`WHERE indexedField = 123` ou`WHERE indexedField IN (456, 789)`.

  QLDB ne respecte pas les inégalités dans les prédicats de requête. Par conséquent, les analyses filtrées par plage de valeurs ne sont pas mises en œuvre.
- Les noms de champs indexés sont sensibles à la casse et peuvent contenir un maximum de 128 caractères.
- La création d'index dans QLDB est asynchrone. Le temps nécessaire à la création d'un index sur une table non vide varie en fonction de la taille de la table. Pour plus d'informations, veuillez consulter Gestion des index.

Description des index

La création d'index dans QLDB est asynchrone. Le temps nécessaire à la création d'un index sur une table non vide varie en fonction de la taille de la table. Pour vérifier l'état de la génération d'un index, vous pouvez interroger la table du catalogue système information_schema.user_tables.

Par exemple, l'instruction suivante interroge le catalogue système pour tous les index de la`VehicleRegistration` table.

```
SELECT VALUE indexes
FROM information_schema.user_tables info, info.indexes indexes
WHERE info.name = 'VehicleRegistration'
```

```
{
    indexId: "Djg2nt0yIs2GY0T29Kud1z",
    expr: "[VIN]",
    status: "ONLINE"
},
{
    indexId: "4tPW3fUhaVhDinRgKRLhGU",
    expr: "[LicensePlateNumber]",
    status: "FAILED",
    message: "aws.ledger.errors.InvalidEntityError: Document contains multiple values
 for indexed field: LicensePlateNumber"
}
```

Champs d'index

- `indexId`— ID unique de l'index.
- `expr`— Le chemin du document indexé. Ce champ est une chaîne de la forme :`[fieldName]`.
- `status`— État actuel de l'index. L'état d'un index peut avoir l'une des valeurs suivantes :
  - `BUILDING`— Construit activement l'index de la table.
  - `FINALIZING`— A fini de créer l'index et commence à l'activer pour utilisation.
  - `ONLINE`— Est actif et prêt à être utilisé dans les requêtes. QLDB n'utilise pas l'index dans les requêtes tant que le statut n'est pas en ligne.
  - `FAILED`— Impossible de créer l'index en raison d'une erreur irrécupérable. Les index dans cet état sont toujours pris en compte dans votre quota d'index par table. Pour plus d'informations, veuillez consulter Erreurs courantes.
  - `DELETING`— Supprime activement l'index après qu'un utilisateur l'ait supprimé.
- `message`— Le message d'erreur qui décrit la raison pour laquelle l'index possède un`FAILED` statut. Ce champ est inclus uniquement pour les index défaillants.

Utilisation de la console

Vous pouvez également utiliserAWS Management Console pour vérifier l'état d'un index.

Pour vérifier l'état d'un index (console)

1. Connectez-vous au et ouvrezAWS Management Console la console Amazon QLDB à l'adresse https://console.aws.amazon.com/qldb.
2. Dans le panneau de navigation, Ledgers.
3. Dans la liste des livres, choisissez le nom du livre dont vous souhaitez gérer les index.
4. Sur la page des détails du registre, sous l'onglet Tables, choisissez le nom de la table dont vous souhaitez vérifier l'index.
5. Sur la page des détails du tableau, recherchez la fiche Champs indexés. La colonne État de l'index affiche l'état actuel de chaque index de la table.

Supprimer des index

Utilisez l'DROP INDEXinstruction pour supprimer un index. Lorsque vous supprimez un index, il est définitivement supprimé de la table.

Tout d'abord, recherchez l'ID d'index à partir de`information_schema.user_tables`. Par exemple, la requête suivante renvoie`indexId` le`LicensePlateNumber` champ indexé de la`VehicleRegistration` table.

```
SELECT indexes.indexId
FROM information_schema.user_tables info, info.indexes indexes
WHERE info.name = 'VehicleRegistration' and indexes.expr = '[LicensePlateNumber]'
```

Utilisez ensuite cet ID pour supprimer l'index. L'exemple suivant montre comment l'ID d'index est supprimé`4tPW3fUhaVhDinRgKRLhGU`. ID d'index est un ID unique qui doit être entouré de guillemets doubles.

```
DROP INDEX "4tPW3fUhaVhDinRgKRLhGU" ON VehicleRegistration WITH (purge = true)
```

> **Note**
>
> La clause`WITH (purge = true)` est obligatoire pour toutes les`DROP INDEX` instructions et`true` est actuellement la seule valeur prise en charge.

Le mot clé`purge` distingue majuscules et minuscules et doit être entièrement en minuscules.

Utilisation de la console

Vous pouvez également utiliser leAWS Management Console pour supprimer un index.

Pour supprimer un index (console)

1. Connectez-vous au et ouvrezAWS Management Console la console Amazon QLDB à l'adresse https://console.aws.amazon.com/qldb.
2. Dans le panneau de navigation, Ledgers.
3. Dans la liste des livres, choisissez le nom du livre dont vous souhaitez gérer les index.
4. Sur la page des détails du registre, sous l'onglet Tables, choisissez le nom de la table dont vous souhaitez supprimer l'index.
5. Sur la page des détails du tableau, recherchez la fiche Champs indexés. Sélectionnez l'index que vous souhaitez supprimer, puis choisissez Supprimer l'index.

Erreurs courantes

Cette section décrit les erreurs courantes que vous pouvez rencontrer lors de la création d'index et suggère des solutions possibles.

> **Note**
>
> Les index dont le statut est sont`FAILED` toujours pris en compte dans votre quota d'index par table. Un index défaillant vous empêche également de modifier ou de supprimer les documents à l'origine de l'échec de la création de l'index dans la table.
> Vous devez supprimer explicitement l'index pour le supprimer du quota.

Le document contient plusieurs valeurs pour le champ indexé : ***`fieldName`***.

QLDB n'est pas en mesure de créer un index pour le nom de champ spécifié car la table contient un document contenant plusieurs valeurs pour le même champ (c'est-à-dire des noms de champ dupliqués).

Vous devez d'abord supprimer l'index défaillant. Assurez-vous ensuite que tous les documents de la table n'ont qu'une seule valeur pour chaque nom de champ avant de réessayer de créer l'index. Vous pouvez également créer un index pour un autre champ qui ne comporte aucun doublon.

QLDB renvoie également cette erreur si vous essayez d'insérer un document contenant plusieurs valeurs pour un champ déjà indexé dans la table.

Limite d'index dépassée : Table ***`TableName`*** ne possède déjà ***`aucun`*** index et ne peut pas en créer d'autres.

QLDB impose une limite de cinq index par table, y compris les index défaillants. Vous devez supprimer un index existant avant d'en créer un nouveau.

Aucun index défini avec identifiant : ***`IndexId`***.

Vous avez essayé de supprimer un index qui n'existe pas pour la combinaison d'ID de table et d'index spécifiée. Pour savoir comment vérifier les index existants, consultezDescription des index.

Identifiants uniques dans Amazon QLDB

Cette section décrit les propriétés et les directives d'utilisation des identifiants uniques attribués par le système dans Amazon QLDB. Il fournit également quelques exemples d'identifiants uniques QLDB.

Rubriques

- Propriétés
- Utilisation
- Exemples

Propriétés

Tous les identifiants attribués par le système dans QLDB sont des identifiants uniques universels (UUID). Chaque ID a les propriétés suivantes :

- Numéro UUID 128 bits
- Représenté dans du texte codé en Base62
- Chaîne alphanumérique de longueur fixe de 22 caractères (par exemple :`3Qv67yjXEwB9SjmvkuG6Cp`)

Utilisation

Lorsque vous utilisez des identifiants uniques QLDB dans votre application, tenez compte des directives suivantes :

Faire

- Traitez l'identifiant comme une chaîne.

Ne pas

- Essayez de décoder la chaîne.
- Attribuez une signification sémantique à la chaîne (par exemple, la dérivation d'une composante temporelle).
- Triez les chaînes dans un ordre sémantique.

Exemples

Les attributs suivants sont des exemples d'identifiants uniques dans QLDB :

- Numéro du document
- Identifiant de l'index
- ID de plage
- ID de table
- Numéro de transaction

Modèle de mise QLDB simultansimultansimultansimultansimultansimultansimultansimultansimult

Amazon QLDB est destiné à répondre aux besoins de travail de travail de travail de travail de traitement de traitement de traitement de traitement de traitement de traitement de traitement de traitement de traitement de traitement de traitement de traitement de traitement de traitement de QLDB prend en charge des fonctionnalités de requête de type SQL et fournit des transactions ACID complètes. De plus, les éléments de données QLDB sont des documents qui offrent une flexibilité de schéma et une modélisation intuitive des données. Grâce à un journal, vous pouvez utiliser QLDB pour accéder à l'historique complet et vérifiable de toutes les modifications apportées à vos données et diffuser des transactions cohérentes vers d'autres services de données selon vos besoins.

Rubriques

Contrôle optimiste de la simultanéité

Dans QLDB, le contrôle de la concurrence est mis en œuvre à l'aide d'un contrôle de concurrence optimiste (OCC). L'OCC fonctionne selon le principe que plusieurs transactions peuvent souvent être effectuées sans interférer les unes avec les autres.

Grâce à l'OCC, les transactions dans QLDB ne verrouillent pas les ressources de la base de données et fonctionnent avec une isolation sérialisable complète. QLDB exécute des transactions simultanées en série, de telle sorte qu'il produit le même effet que si ces transactions étaient lancées en série.

Avant de valider, chaque transaction effectue un contrôle de validation pour s'assurer qu'aucune autre transaction validée n'a modifié les données auxquelles elle accède. Si cette vérification révèle des modifications contradictoires ou si l'état des données change, la transaction de validation est rejetée. Toutefois, la transaction peut être redémarrée.

Lorsqu'une transaction est écrite dans QLDB, les contrôles de validation du modèle OCC sont implémentés par QLDB lui-même. Si une transaction ne peut pas être écrite dans le journal en raison d'un échec lors de la phase de vérification de l'OCC, QLDB renvoie une transaction`OccConflictException` à la couche application. Le logiciel d'application est chargé de s'assurer que la transaction est redémarrée. L'application doit abandonner la transaction refusée, puis réessayer l'ensemble de la transaction depuis le début.

Pour savoir comment le pilote QLDB gère et réessaie les conflits OCC et les autres exceptions transitoires, consultezComprendre la stratégie de nouvelle tentative avec le pilote dans Amazon QLDB.

Utilisation d'index pour éviter l'analyse complète des tables

Dans QLDB, chaque instruction partiQL (y compris chaque`SELECT` requête) est traitée dans le cadre d'une transaction et est soumise à un délai d'expiration de transaction.

Il est recommandé d'exécuter des instructions avec une clause de`WHERE` prédicat qui filtre en fonction d'un champ indexé ou d'un identifiant de document. QLDB a besoin d'un opérateur d'égalité sur un champ indexé pour rechercher efficacement un document ; par exemple,`WHERE indexedField = 123` ou`WHERE indexedField IN (456, 789)`.

Sans cette recherche indexée, QLDB doit effectuer une analyse complète de la table lors de la lecture de documents. Cela peut entraîner une latence des requêtes et des délais de transaction, et augmente également les risques de conflit OCC avec des transactions concurrentes.

Par exemple, considérez une table nommée`Vehicle` dont le`VIN` champ ne comporte qu'un index. Il contient les documents suivants.

VIN	Marque	Modèle	Couleur
`"1N4AL11D 75C109151"`	`"Audi"`	`"A5"`	`"Silver"`
`"KM8SRDHF 6EU074761"`	`"Tesla"`	`"Model S"`	`"Blue"`
`"3HGGK5G5 3FM761765"`	`"Ducati"`	`"Monster 1200"`	`"Yellow"`

VIN	Marque	Modèle	Couleur
"1HVBBAAN XWH544237"	"Ford"	"F 150"	"Black"
"1C4RJFAG 0FC625797"	"Mercedes"	"CLK 350"	"White"

Deux utilisateurs simultanés nommés Alice et Bob travaillent avec la même table dans un registre. Ils souhaitent mettre à jour deux documents différents, comme suit.

Alice :

```
UPDATE Vehicle AS v
SET v.Color = 'Blue'
WHERE v.VIN = '1N4AL11D75C109151'
```

Bob :

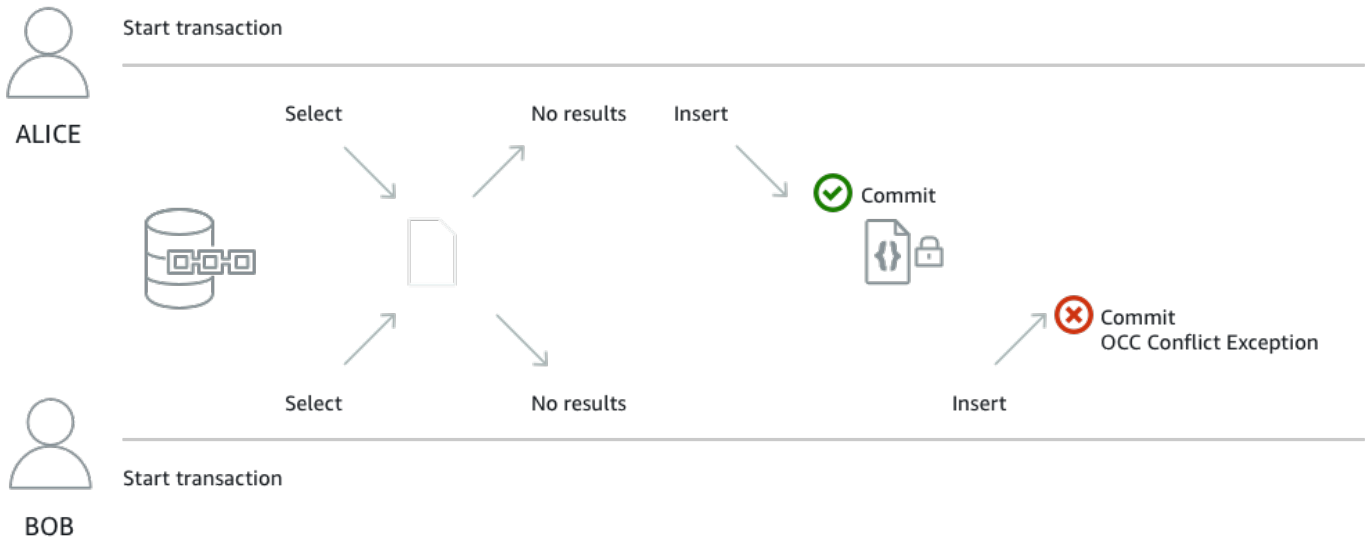
```
UPDATE Vehicle AS v
SET v.Color = 'Red'
WHERE v.Make = 'Tesla' AND v.Model = 'Model S'
```

Supposons qu'Alice et Bob commencent leurs transactions en même temps. La`UPDATE` déclaration d'Alice effectue une recherche indexée sur le`VIN` champ, elle n'a donc besoin que de lire ce document. Alice termine et valide sa transaction avec succès en premier.

L'instruction de Bob filtre les champs non indexés. Elle analyse donc une table et trouve un`OccConflictException`. Cela est dû au fait que la transaction validée d'Alice a modifié les données auxquelles la déclaration de Bob accède, ce qui inclut tous les documents du tableau, et pas seulement le document que Bob est en train de mettre à jour.

Conflits d'insertion OCC

Les conflits OCC peuvent inclure des documents récemment insérés, et pas uniquement des documents qui existaient auparavant. Examinez le diagramme suivant, dans lequel deux utilisateurs simultanéité (Alice et Bob) utilisent la même table dans un registre. Ils souhaitent tous deux insérer un nouveau document uniquement à la condition qu'aucune valeur de prédicat n'existe encore.



Dans cet exemple, Alice et Bob exécutent les`INSERT` instructions`SELECT` et les instructions suivantes au sein d'une seule transaction. Leur application exécute l'`INSERT`instruction uniquement si celle-ci`SELECT` ne donne aucun résultat.

```
SELECT * FROM Vehicle v WHERE v.VIN = 'ABCDE12345EXAMPLE'
```

```
INSERT INTO Vehicle VALUE
{
    'VIN' : 'ABCDE12345EXAMPLE',
    'Type' : 'Wagon',
    'Year' : 2019,
    'Make' : 'Subaru',
    'Model' : 'Outback',
    'Color' : 'Gray'
}
```

Supposons qu'Alice et Bob commencent leurs transactions en même temps. Leurs deux`SELECT` requêtes ne renvoient aucun document existant avec un`VIN` of`ABCDE12345EXAMPLE`. Leurs demandes se poursuivent donc avec la`INSERT` déclaration.

Alice termine et valide sa transaction avec succès en premier. Ensuite, Bob essaie de valider sa transaction, mais QLDB la rejette et lance un`OccConflictException`. Cela est dû au fait que la transaction validée d'Alice a modifié le jeu de résultats de la`SELECT` requête de Bob et qu'OCC détecte ce conflit avant de valider la transaction de Bob.

La`SELECT` requête est requise pour que cet exemple de transaction soit idempotent. Bob peut alors recommencer l'ensemble de sa transaction depuis le début. Mais sa prochaine`SELECT` requête renverra le document qu'Alice a inséré, de sorte que l'application de Bob n'exécutera pas le`INSERT`.

Capacités de mise à l'échelle de mise à

La transaction d'insertion présentée dans la section précédente est également un exemple de transaction idempotente. En d'autres termes, l'exécution de la même transaction plusieurs fois produit des résultats identiques. Si Bob exécute le`INSERT` sans vérifier au préalable si un élément existe`VIN` déjà, le tableau peut se retrouver avec des documents contenant des`VIN` valeurs dupliquées.

Outre les conflits OCC, envisagez d'autres scénarios de nouvelle tentative. Par exemple, il est possible que QLDB valide avec succès une transaction côté serveur, mais que le client expire en attendant une réponse. Il est recommandé de rendre vos transactions d'écriture idempotentes afin d'éviter tout effet secondaire inattendu en cas de simultanéité ou de nouvelles tentatives.

Conflits de rédaction OCC

QLDB empêche les rédactions simultanées de révisions sur le même bloc de journal. Prenons l'exemple de deux utilisateurs simultanés (Alice et Bob) qui souhaitent rédiger deux révisions de documents différentes qui sont validées sur le même bloc dans un registre. Tout d'abord, Alice demande la rédaction d'une révision en exécutant la procédure`REDACT_REVISION` stockée, comme suit.

```
EXEC REDACT_REVISION `{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`,
 '5PLf9SXwndd63lPaSIa0O6', 'ADR2Ll1fGsU4Jr4EqTdnQF'
```

Ensuite, alors que la demande d'Alice est toujours en cours de traitement, Bob demande la rédaction d'une autre révision, comme suit.

```
EXEC REDACT_REVISION `{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`,
 '8F0TPCmdNQ6JTRpiLj2TmW', '05K8zpGYWynDlEOK5afDRc'
```

QLDB rejette la demande de Bob`OccConflictException` même s'ils essaient de rédiger deux révisions de documents différentes. Cela est dû au fait que la révision de Bob se trouve dans le même bloc que la révision qu'Alice est en train de rédiger. Une fois le traitement de la demande d'Alice terminé, Bob peut réessayer sa demande de rédaction.

De même, si deux transactions simultanées tentent de modifier la même révision, une seule demande peut être traitée. L'autre demande échoue avec une exception de conflit OCC jusqu'à ce que la rédaction soit terminée. Par la suite, toute demande de rédaction de la même révision entraînera une erreur indiquant que la révision est déjà expurgée.

Gestion de mise à simultanéité

Si vous avez une expérience concernant l'utilisation d'un système de gestion de base de gestion de base de gestion de base de base de gestion de base de gestion de base de gestion de base de gestion de base de gestion de base de gestion de base de gestion de base de gestion de base de gestion de base QLDB n'a pas le même concept de connexion RDBMS traditionnelle, car les transactions sont exécutées à l'aide de messages de requête et de réponse HTTP.

Dans QLDB, le concept analogue est celui d'une session active. Une session est conceptuellement similaire à une connexion utilisateur : elle gère les informations relatives à vos demandes de transactions de données adressées à un registre. Une session active est une session qui exécute activement une transaction. Il peut également s'agir d'une session qui a récemment terminé une transaction et où le service prévoit d'en démarrer une autre immédiatement. QLDB prend en charge une transaction active par session.

La limite de sessions actives simultanées par registre est définie dansQuotas et limites d'Amazon QLDB. Une fois cette limite atteinte, toute session qui tente de démarrer une transaction génère une erreur (`LimitExceededException`).

Pour plus d'informations sur le cycle de vie d'une session et sur la manière dont le pilote QLDB gère les sessions lors de l'exécution de transactions de données, consultezGestion des sessions avec le chauffeur. Pour connaître les meilleures pratiques relatives à la configuration d'un pool de sessions dans votre application à l'aide du pilote QLDB, consultez les recommandations relativesConfiguration de QldbDriver objet au pilote Amazon QLDB.

Vérification des données dans Amazon QLDB

Avec Amazon QLDB, vous pouvez être sûr que l'historique des modifications apportées aux données de votre application est exact. QLDB utilise un journal transactionnel immuable, appelé journal, pour le stockage des données. Le journal suit chaque modification apportée à vos données enregistrées et conserve un historique complet et vérifiable des modifications au fil du temps.

QLDB utilise la fonction de hachage SHA-256 avec un modèle basé sur l'arbre de Merkle pour générer une représentation cryptographique de votre journal, connue sous le nom de résumé. Le condensé agit comme une signature unique de l'historique complet des modifications de vos données à un moment donné. Vous utilisez le résumé pour vérifier l'intégrité des révisions de votre document par rapport à cette signature.

Rubriques

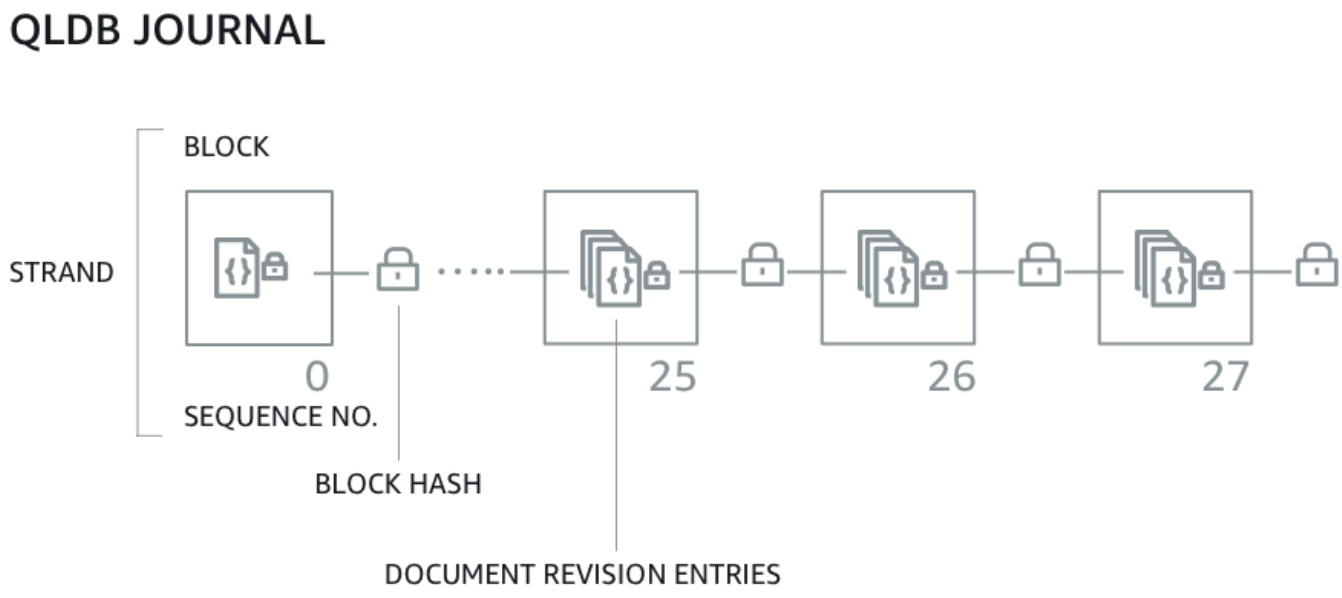
Quel type de données pouvez-vous vérifier dans QLDB ?

Dans QLDB, chaque registre possède exactement un journal. Un journal peut comporter plusieurs volets, qui sont des partitions du journal.

> **Note**
>
> QLDB prend actuellement en charge les revues à un seul volet.

Un bloc est un objet qui est inscrit dans le volet journal lors d'une transaction. Ce bloc contient des objets d'entrée, qui représentent les révisions du document résultant de la transaction. Vous pouvez vérifier une révision individuelle ou un bloc de journal complet dans QLDB.

Le schéma suivant illustre cette structure de journal.



Le diagramme montre que les transactions sont enregistrées dans le journal sous forme de blocs contenant des entrées de révision de documents. Cela montre également que chaque bloc est enchaîné par hachage aux blocs suivants et possède un numéro de séquence pour spécifier son adresse dans le brin.

Pour plus d'informations sur le contenu des données d'un bloc, consultezContenu du journal dans Amazon QLDB.

Que signifie l'intégrité des données ?

L'intégrité des données dans QLDB signifie que le journal de votre registre est en fait immuable. En d'autres termes, vos données (en particulier, chaque révision de document) sont dans un état dans lequel les conditions suivantes sont vraies :

1. Il se trouve au même endroit dans votre journal où il a été écrit pour la première fois.
2. Il n'a subi aucune modification depuis qu'il a été écrit.

Comment fonctionne la vérification ?

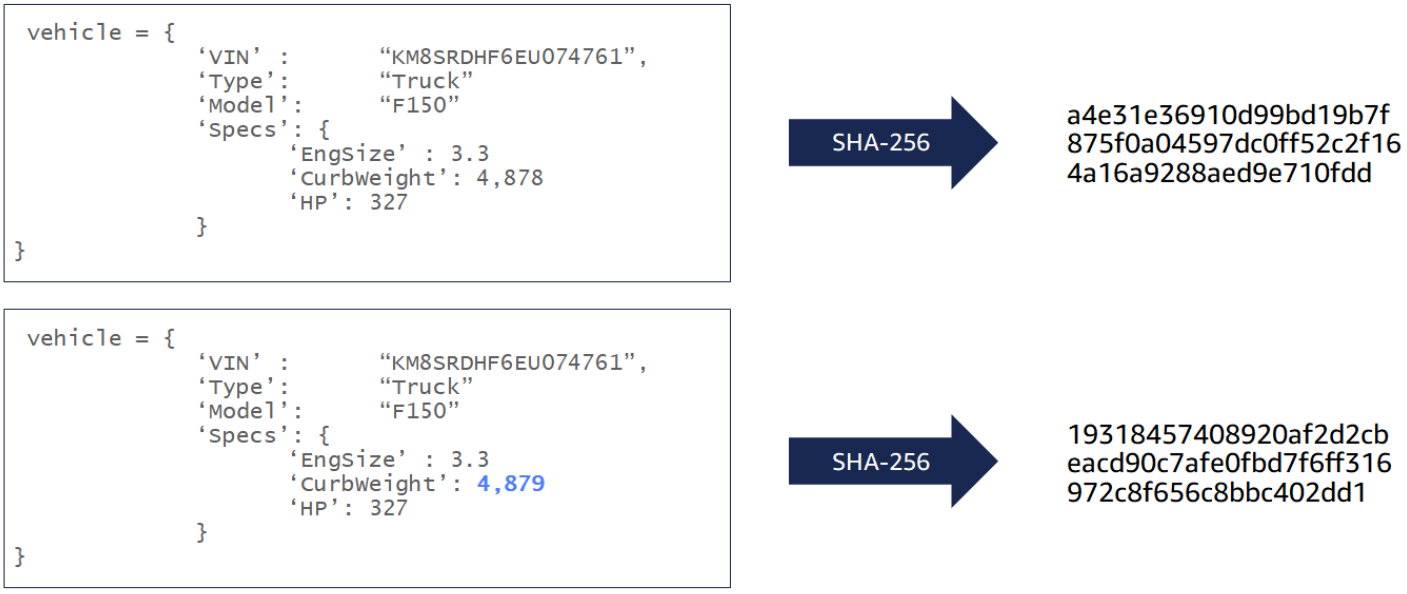
Pour comprendre le fonctionnement de la vérification dans Amazon QLDB, vous pouvez décomposer le concept en quatre éléments de base.

- Hachage
- Digest
- Arbre Merkle
- Preuve

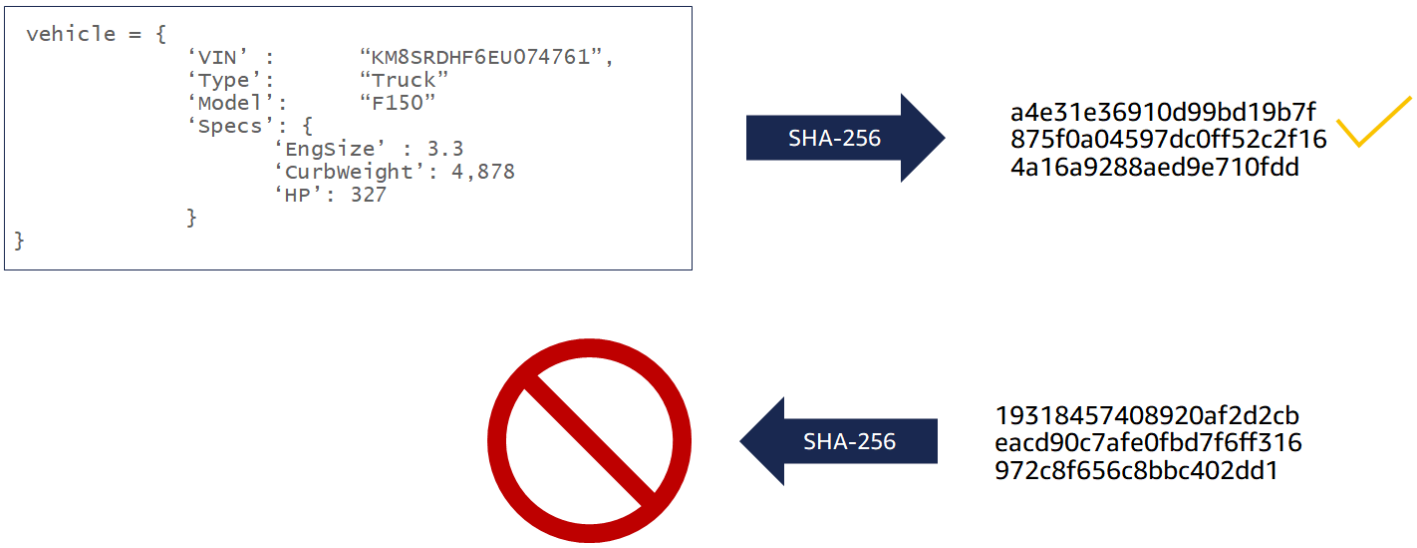
Hachage

QLDB utilise la fonction de hachage cryptographique SHA-256 pour créer des valeurs de hachage de 256 bits. Un hachage agit comme une signature unique et de longueur fixe de toute quantité arbitraire de données d'entrée. Si vous modifiez une partie de l'entrée, ne serait-ce qu'un seul caractère ou un seul bit, le hachage de sortie change complètement.

Le schéma suivant montre que la fonction de hachage SHA-256 crée des valeurs de hachage totalement uniques pour deux documents QLDB qui ne diffèrent que d'un chiffre.



La fonction de hachage SHA-256 est unidirectionnelle, ce qui signifie qu'il n'est pas mathématiquement possible de calculer l'entrée lorsqu'une sortie est donnée. Le schéma suivant montre qu'il n'est pas possible de calculer le document QLDB d'entrée lorsqu'on lui donne une valeur de hachage en sortie.



Les entrées de données suivantes sont hachées dans QLDB à des fins de vérification :

- Révisions du document
- Instructions PartiQL
- Entrées de révision
- Blocs de journaux

Digest

Un résumé est une représentation cryptographique de l'intégralité du journal de votre registre à un moment donné. Un journal est uniquement en ajout, et les blocs de journal sont séquencés et hachés de la même manière que les blockchains.

Vous pouvez demander un résumé pour un registre à tout moment. QLDB génère le condensé et vous le renvoie sous forme de fichier de sortie sécurisé. Vous utilisez ensuite ce résumé pour vérifier l'intégrité des révisions de documents qui ont été validées à un moment antérieur. Si vous recalculez les hachages en commençant par une révision et en terminant par le résumé, vous prouvez que vos données n'ont pas été modifiées entre les deux.

Arbre Merkle

À mesure que la taille de votre registre augmente, il devient de plus en plus inefficace de recalculer la chaîne de hachage complète du journal à des fins de vérification. QLDB utilise un modèle d'arbre Merkle pour remédier à cette inefficacité.

Un arbre Merkle est une structure de données arborescente dans laquelle chaque nœud de feuille représente le hachage d'un bloc de données. Chaque nœud autre qu'une feuille est un hachage de ses nœuds enfants. Couramment utilisé dans les chaînes de blocs, un arbre Merkle vous aide à vérifier efficacement de grands ensembles de données grâce à un mécanisme de preuve d'audit. Pour plus d'informations sur les arbres Merkle, consultez la page Wikipédia sur les arbres Merkle. Pour en savoir plus sur les preuves d'audit Merkle et pour un exemple de cas d'utilisation, consultez How Log Proofs Work sur le site de transparence des certificats.

L'implémentation QLDB de l'arbre Merkle est construite à partir de la chaîne de hachage complète d'un journal. Dans ce modèle, les nœuds foliaires sont l'ensemble de tous les hachages de révision de documents individuels. Le nœud racine représente le résumé de l'intégralité du journal à un moment donné.

À l'aide d'une preuve d'audit Merkle, vous pouvez vérifier une révision en vérifiant uniquement un petit sous-ensemble de l'historique des révisions de votre registre. Pour ce faire, vous devez parcourir l'arbre depuis un nœud de feuille donné (révision) jusqu'à sa racine (résumé). Le long de ce chemin de traversée, vous hachez de manière récursive des paires de nœuds frères pour calculer leur hachage parent jusqu'à ce que vous terminiez avec le condensé. Cette traversée a une complexité temporelle en ce qui concerne `log(n)` les nœuds de l'arbre.

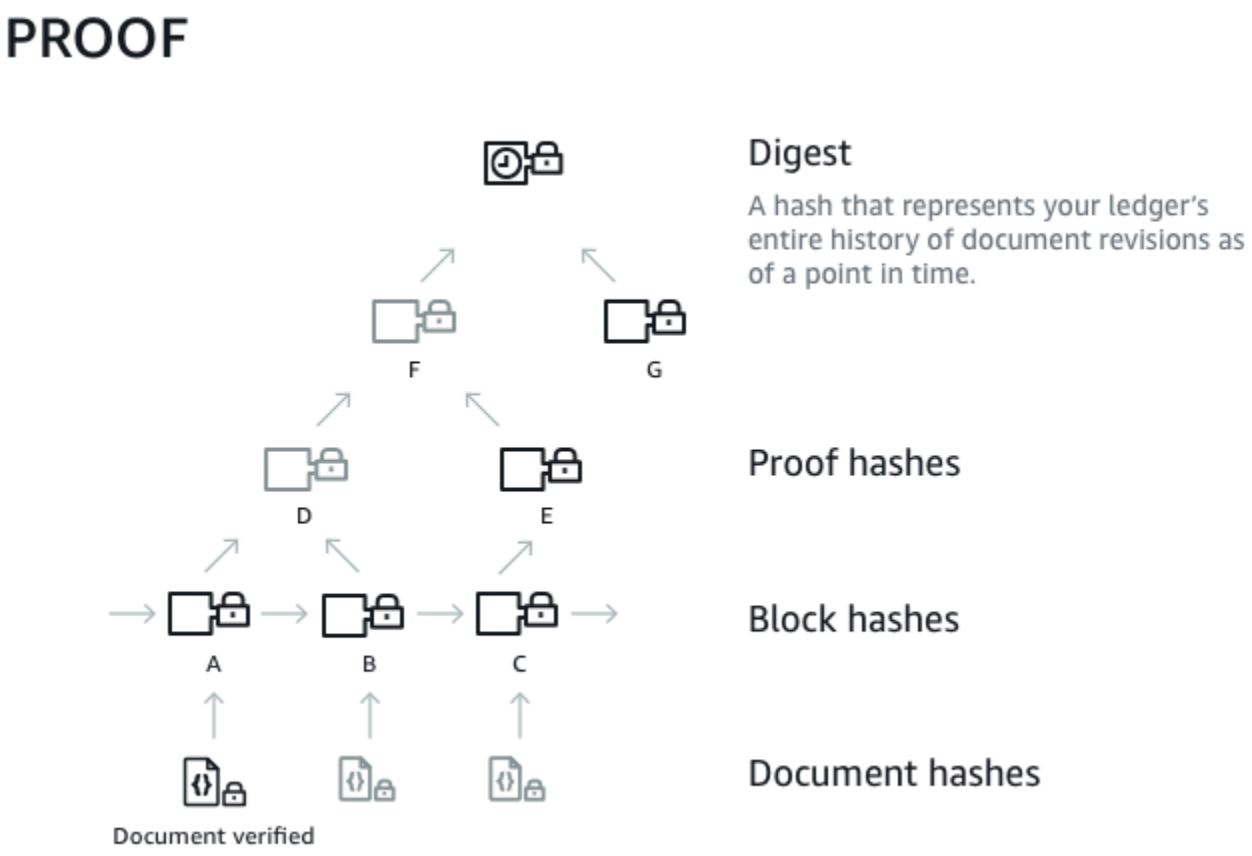
Preuve

La preuve en est la liste ordonnée des hachages de nœuds que QLDB renvoie pour un condensé et une révision de document donnés. Il comprend les hachages requis par un modèle d'arbre Merkle pour enchaîner le hachage du nœud feuille donné (une révision) au hachage racine (le condensé).

La modification de données validées entre une révision et un résumé rompt la chaîne de hachage de votre journal et rend impossible la génération d'une preuve.

Exemple de vérification

Le schéma suivant illustre le modèle d'arbre de hachage Amazon QLDB. Il montre un ensemble de hachages de blocs qui s'enroulent jusqu'au nœud racine supérieur, qui représente le condensé d'un fil de journal. Dans un registre comportant un journal à un seul brin, ce nœud racine est également le condensé de l'ensemble du registre.



Supposons que le nœud A soit le bloc contenant la révision du document dont vous souhaitez vérifier le hachage. Les nœuds suivants représentent la liste ordonnée des hachages que QLDB fournit dans votre preuve : B, E, G. Ces hachages sont nécessaires pour recalculer le condensé à partir du hachage A.

Pour recalculer le résumé, procédez comme suit :

1. Commencez par le hachage A et concaténez-le avec le hachage B. Ensuite, hachez le résultat pour calculer D.
2. Utilisez D et E pour calculer F.
3. Utilisez F et G pour calculer le condensé.

La vérification est réussie si votre résumé recalculé correspond à la valeur attendue. À partir d'un hachage de révision et d'un résumé, il n'est pas possible de rétroconcevoir les hachages dans une

preuve. Par conséquent, cet exercice prouve que votre révision a bien été écrite à cet emplacement de journal par rapport au condensé.

Quel est l'impact de la suppression des données sur la vérification ?

Dans Amazon QLDB, `DELETE` une instruction ne supprime logiquement un document qu'en créant une nouvelle révision qui le marque comme supprimé. QLDB prend également en charge une opération de rédaction de données qui vous permet de supprimer définitivement les révisions de document inactives dans l'historique d'un tableau.

L'opération de rédaction supprime uniquement les données utilisateur dans la révision spécifiée et laisse inchangées la séquence du journal et les métadonnées du document. Une fois qu'une révision est expurgée, les données utilisateur de la révision (représentées par la `data` structure) sont remplacées par un nouveau `dataHash` champ. La valeur de ce champ est le hachage Amazon Ion de la `data` structure supprimée. Pour plus d'informations et un exemple d'opération de rédaction, consultezRédaction de révisions de documents.

Par conséquent, le registre conserve l'intégrité globale de ses données et reste vérifiable cryptographiquement grâce aux opérations d'API de vérification existantes. Vous pouvez toujours utiliser ces opérations d'API comme prévu pour demander un condensé (GetDigest), demander une preuve (GetBlockou GetRevision), puis exécuter votre algorithme de vérification à l'aide des objets renvoyés.

Recalculer un hachage de révision

Si vous envisagez de vérifier une révision individuelle d'un document en recalculant son hachage, vous devez vérifier de manière conditionnelle si la révision a été expurgée. Si la révision a été supprimée, vous pouvez utiliser la valeur de hachage fournie dans le `dataHash` champ. S'il n'a pas été expurgé, vous pouvez recalculer le hachage en utilisant le champ. `data`

En effectuant cette vérification conditionnelle, vous pouvez identifier les révisions expurgées et prendre les mesures appropriées. Par exemple, vous pouvez enregistrer les événements de manipulation de données à des fins de surveillance.

Commencer à utiliser la vérification

Avant de pouvoir vérifier les données, vous devez demander un résumé à partir de votre registre et l'enregistrer pour plus tard. Toute révision de document validée avant le dernier bloc couvert par le résumé peut être vérifiée par rapport à ce résumé.

Ensuite, vous demandez une preuve à Amazon QLDB pour une révision éligible que vous souhaitez vérifier. À l'aide de cette preuve, vous appelez une API côté client pour recalculer le résumé, en commençant par le hachage de votre révision. Tant que le résumé précédemment enregistré est connu et fiable en dehors de QLDB, l'intégrité de votre document est prouvée si le hachage du résumé recalculé correspond au hachage du résumé enregistré.

> **Important**
>
> - Ce que vous prouvez spécifiquement, c'est que la révision du document n'a pas été modifiée entre le moment où vous avez enregistré ce résumé et celui où vous avez effectué la vérification. Vous pouvez demander et enregistrer un résumé dès qu'une révision que vous souhaitez vérifier ultérieurement est validée dans le journal.
> - À titre de bonne pratique, nous vous recommandons de demander régulièrement des résumés et de les stocker en dehors du registre. Déterminez la fréquence à laquelle vous demandez des résumés en fonction de la fréquence à laquelle vous validez les révisions dans votre registre.
>
> Pour un article de AWS blog détaillé qui aborde la valeur de la vérification cryptographique dans le contexte d'un cas d'utilisation réaliste, consultez la section Vérification cryptographique dans le monde réel avec Amazon QLDB.

Pour obtenir step-by-step des guides sur la façon de demander un résumé à partir de votre registre, puis de vérifier vos données, consultez ce qui suit :

- Étape 1 : Demande d'un résumé dans QLDB
- Étape 2 : vérification de vos données dans QLDB

Étape 1 : Demande d'un résumé dans QLDB

Amazon QLDB fournit une API permettant de demander un résumé reprenant l'extrait actuel du journal figurant dans votre registre. Le conseil du journal fait référence au dernier bloc validé au moment où QLDB reçoit votre demande. Vous pouvez utiliser le AWS Management Console, un AWS SDK ou le AWS Command Line Interface (AWS CLI) pour obtenir un résumé.

Rubriques

AWS Management Console

Procédez comme suit pour demander un résumé à l'aide de la console QLDB.

Pour demander un résumé (console)

1. Connectez-vous à la AWS Management Console console Amazon QLDB et ouvrez-la à l'adresse https://console.aws.amazon.com/qldb.
2. Dans le volet de navigation, choisissez Ledgers.
3. Dans la liste des registres, sélectionnez le nom du registre pour lequel vous souhaitez demander un résumé.
4. Choisissez Get digest. La boîte de dialogue Obtenir le résumé affiche les détails suivants :
   - Résumé — La valeur de hachage SHA-256 du résumé que vous avez demandé.
   - Adresse du résumé : emplacement du dernier bloc dans le journal concerné par le résumé que vous avez demandé. Une adresse comporte les deux champs suivants :
     - `strandId`— L'identifiant unique du volet du journal qui contient le bloc.
     - `sequenceNo`— Le numéro d'index qui indique l'emplacement du bloc dans le fil.
   - Grand livre : nom du registre pour lequel vous avez demandé un résumé.
   - Date — Horodatage auquel vous avez demandé le résumé.
5. Passez en revue les informations du résumé. Ensuite, choisissez Save (Enregistrer). Vous pouvez conserver le nom de fichier par défaut ou en saisir un nouveau.

> **Note**
>
> Vous remarquerez peut-être que les valeurs de hachage de votre résumé et de votre adresse de pourboire changent même si vous ne modifiez aucune donnée de votre registre. Cela est dû au fait que la console récupère le catalogue système du registre chaque fois que vous exécutez une requête dans l'éditeur partiQL. Il s'agit d'une transaction de lecture qui est validée dans le journal et entraîne la modification de la dernière adresse de bloc.

Cette étape enregistre un fichier texte brut dont le contenu est au format Amazon Ion. Le fichier porte l'extension de nom de fichier `.ion.txt` et contient toutes les informations de synthèse répertoriées dans la boîte de dialogue précédente. Voici un exemple du contenu d'un fichier de synthèse. L'ordre des champs peut varier en fonction de votre navigateur.

```
{
  "digest": "42zaJOfV8iGutVGNaIuzQWhD5Xb/5B9lScHnvxPXm9E=",
  "digestTipAddress": "{strandId:\"BlFTjlSXze9BIh1KOszcE3\",sequenceNo:73}",
  "ledger": "my-ledger",
  "date": "2019-04-17T16:57:26.749Z"
}
```

6. Enregistrez ce fichier pour pouvoir y accéder à l'avenir. Plus tard, vous pourrez utiliser ce fichier pour vérifier une révision de document par rapport à.

> **Important**
>
> La révision du document que vous vérifierez ultérieurement doit être couverte par le résumé que vous avez enregistré. En d'autres termes, le numéro de séquence de l'adresse du document doit être inférieur ou égal au numéro de séquence de l'adresse du Digest tip.

API QLDB

Vous pouvez également demander un résumé à partir de votre registre en utilisant l'API Amazon QLDB avec AWS un SDK ou le. AWS CLI L'API QLDB fournit les opérations suivantes destinées à être utilisées par les programmes d'application :

- GetDigest— Renvoie le résumé d'un registre au dernier bloc validé du journal. La réponse inclut une valeur de hachage de 256 bits et une adresse de bloc.

Pour plus d'informations sur la demande d'un résumé à l'aide de AWS CLI, consultez la commande get-digest dans la référence des AWS CLI commandes.

Exemple d'application

Pour des exemples de code Java, consultez le GitHub référentiel amazon-qldb-dmv-sampleaws-samples/ -java. Pour obtenir des instructions sur le téléchargement et l'installation de cet exemple d'application, consultezInstallation de l'exemple d'application Java Amazon QLDB. Avant de demander un résumé, assurez-vous de suivre les étapes 1 à 3 Tutoriel Java pour créer un registre d'échantillons et le charger avec des exemples de données.

Le code du didacticiel présenté en classe GetDigestfournit un exemple de demande de résumé à partir du registre `vehicle-registration` d'échantillons.

Pour vérifier la révision d'un document à l'aide du résumé que vous avez enregistré, passez àÉtape 2 : vérification de vos données dans QLDB.

Étape 2 : vérification de vos données dans QLDB

Amazon QLDB fournit une API permettant de demander une preuve pour un identifiant de document spécifié et le bloc associé. Vous devez également fournir l'adresse du conseil d'un résumé que vous avez précédemment enregistré, comme décrit dansÉtape 1 : Demande d'un résumé dans QLDB. Vous pouvez utiliser le AWS Management Console, un AWS SDK ou le AWS CLI pour obtenir une preuve.

Vous pouvez ensuite utiliser la preuve renvoyée par QLDB pour vérifier la révision du document par rapport au résumé enregistré, à l'aide d'une API côté client. Cela vous permet de contrôler l'algorithme que vous utilisez pour vérifier vos données.

Rubriques

- AWS Management Console
- API QLDB

AWS Management Console

Cette section décrit les étapes permettant de vérifier la révision d'un document par rapport à un résumé enregistré précédemment à l'aide de la console Amazon QLDB.

Avant de commencer, assurez-vous de suivre les étapes décrites dansÉtape 1 : Demande d'un résumé dans QLDB. La vérification nécessite un résumé préalablement enregistré qui couvre la révision que vous souhaitez vérifier.

Pour vérifier la révision d'un document (console)

1. Ouvrez la console Amazon QLDB à l'adresse https://console.aws.amazon.com/qldb.
2. Tout d'abord, recherchez dans votre registre `blockAddress` la `id` et la révision que vous souhaitez vérifier. Ces champs sont inclus dans les métadonnées du document, que vous pouvez interroger dans la vue validée.

   Le document `id` est une chaîne d'identification unique attribuée par le système. `blockAddress`Il s'agit d'une structure ionique qui indique l'emplacement du bloc où la révision a été validée.

   Dans le volet de navigation, choisissez l'éditeur PartiQL.
3. Choisissez le nom du registre sous lequel vous souhaitez vérifier une révision.
4. Dans la fenêtre de l'éditeur de requêtes, entrez une `SELECT` instruction dont la syntaxe est la suivante, puis choisissez Exécuter.

   ```
   SELECT metadata.id, blockAddress FROM _ql_committed_table_name
   WHERE criteria
   ```

   Par exemple, la requête suivante renvoie un document à partir de la `VehicleRegistration` table du registre d'exemple créé dansMise en route avec la console Amazon QLDB.

   ```
   SELECT r.metadata.id, r.blockAddress FROM _ql_committed_VehicleRegistration AS r
   WHERE r.data.VIN = 'KM8SRDHF6EU074761'
   ```
5. Copiez et enregistrez les `blockAddress` valeurs `id` et renvoyées par votre requête. Veillez à omettre les guillemets pour le `id` champ. Dans Amazon Ion, les types de données sous forme de chaîne sont délimités par des guillemets doubles. Par exemple, vous devez uniquement copier le texte alphanumérique dans l'extrait de code suivant.

```
"LtMNJYNjSwzBLgf7sLifrG"
```

6. Maintenant qu'une révision de document est sélectionnée, vous pouvez commencer le processus de vérification.

   Dans le volet de navigation, choisissez Verification.

7. Dans le formulaire Vérifier le document, sous Spécifiez le document que vous souhaitez vérifier, entrez les paramètres d'entrée suivants :

   - Ledger — Le registre dans lequel vous souhaitez vérifier une révision.
   - Adresse du bloc : `blockAddress` valeur renvoyée par votre requête à l'étape 4.
   - ID du document : `id` valeur renvoyée par votre requête à l'étape 4.

8. Sous Spécifier le résumé à utiliser pour la vérification, sélectionnez le résumé que vous avez enregistré précédemment en choisissant Choisir le résumé. Si le fichier est valide, tous les champs du résumé sont automatiquement remplis sur votre console. Vous pouvez également copier et coller manuellement les valeurs suivantes directement à partir de votre fichier condensé :

   - Digest — La `digest` valeur de votre fichier de résumé.
   - Adresse du conseil de synthèse : `digestTipAddress` valeur de votre fichier de résumé.

9. Passez en revue les paramètres d'entrée de votre document et résumez, puis choisissez Verify.

   La console automatise deux étapes pour vous :

   a. Demandez une preuve à QLDB pour le document que vous avez spécifié.

   b. Utilisez la preuve renvoyée par QLDB pour appeler une API côté client, qui vérifie la révision de votre document par rapport au résumé fourni. Pour examiner cet algorithme de vérification, consultez la section suivante API QLDB pour télécharger l'exemple de code.

   La console affiche les résultats de votre demande dans la fiche de résultats de vérification. Pour plus d’informations, consultez Résultats de vérification.

API QLDB

Vous pouvez également vérifier la révision d'un document en utilisant l'API Amazon QLDB avec AWS un SDK ou le. AWS CLI L'API QLDB fournit les opérations suivantes destinées à être utilisées par les programmes d'application :

- `GetDigest`— Renvoie le résumé d'un registre au dernier bloc validé du journal. La réponse inclut une valeur de hachage de 256 bits et une adresse de bloc.
- `GetBlock`— Renvoie un objet de type bloc à une adresse spécifiée dans un journal. Renvoie également une preuve du bloc spécifié pour vérification si elle `DigestTipAddress` est fournie.
- `GetRevision`— Renvoie un objet de données de révision pour un ID de document et une adresse de bloc spécifiés. Renvoie également une preuve de la révision spécifiée pour vérification si elle `DigestTipAddress` est fournie.

Pour une description complète de ces opérations d'API, consultez leRéférence d'API Amazon QLDB.

Pour plus d'informations sur la vérification des données à l'aide du AWS CLI, consultez la référence des AWS CLI commandes.

Exemple d'application

Pour des exemples de code Java, consultez le GitHub référentiel amazon-qldb-dmv-sampleaws-samples/ -java. Pour obtenir des instructions sur le téléchargement et l'installation de cet exemple d'application, consultezInstallation de l'exemple d'application Java Amazon QLDB. Avant de procéder à une vérification, assurez-vous de suivre les étapes 1 à 3 Tutoriel Java pour créer un registre d'échantillons et le charger avec des exemples de données.

Le code du didacticiel présenté en classe GetRevisionfournit un exemple de demande de preuve pour une révision de document, puis de vérification de cette révision. Cette classe exécute les étapes suivantes :

1. Demande un nouveau résumé à partir du registre `vehicle-registration` d'échantillons.
2. Demande une preuve pour un exemple de révision de document à partir du `VehicleRegistration` tableau du `vehicle-registration` registre.
3. Vérifie la révision de l'échantillon à l'aide du résumé et de la preuve renvoyés.

Résultats de vérification

Cette section décrit les résultats renvoyés par une demande de vérification des données Amazon QLDB sur le. AWS Management Console Pour obtenir des instructions détaillées sur la procédure à suivre pour soumettre une demande de vérification, consultezÉtape 2 : vérification de vos données dans QLDB.

Sur la page de vérification de la console QLDB, les résultats de votre demande sont affichés dans la carte des résultats de vérification. L'onglet Preuve affiche le contenu de la preuve renvoyée par QLDB pour la révision et le résumé du document que vous avez spécifiés. Il inclut les informations suivantes :

- Hachage de révision : valeur SHA-256 qui représente de manière unique la révision du document que vous êtes en train de vérifier.
- Hashs de preuve : liste ordonnée des hachages fournie par QLDB qui sont utilisés pour recalculer le condensé spécifié. La console commence par le hachage de révision et le combine séquentiellement avec chaque hachage de preuve jusqu'à ce qu'il se termine par un condensé recalculé.

  La liste est réduite par défaut, vous pouvez donc l'étendre pour révéler les valeurs de hachage. Vous pouvez éventuellement essayer vous-même les calculs de hachage en suivant les étapes décrites dansUtiliser une preuve pour recalculer votre résumé.
- Digest calculé — Le hachage résultant de la série de calculs de hachage effectués sur le hachage de révision. Si cette valeur correspond à votre résumé précédemment enregistré, la vérification est réussie.

L'onglet Bloc affiche le contenu du bloc contenant la révision que vous êtes en train de vérifier. Il inclut les informations suivantes :

- ID de transaction — L'identifiant unique de la transaction qui a validé ce bloc.
- Heure de la transaction : date à laquelle ce bloc a été validé dans le volet.
- Hachage du bloc : valeur SHA-256 qui représente de manière unique ce bloc et l'ensemble de son contenu.
- Adresse du bloc : emplacement dans le journal de votre registre où ce blocage a été validé. Une adresse comporte les deux champs suivants :
  - ID du volet — L'identifiant unique du volet du journal qui contient ce bloc.

  - Numéro de séquence : numéro d'index qui indique l'emplacement de ce bloc dans le brin.
- Instructions : instructions partiQL exécutées pour valider les entrées de ce bloc.

> **Note**
>
> Si vous exécutez des instructions paramétrées par programmation, elles sont enregistrées dans vos blocs de journal avec des paramètres de liaison au lieu des données littérales. Par exemple, vous pouvez voir l'instruction suivante dans un bloc de journal, où le point d'interrogation (?) est un espace réservé variable pour le contenu du document.
>
> ```
> INSERT INTO Vehicle ?
> ```

- Entrées de documents : révisions du document validées dans ce bloc.

Si votre demande n'a pas réussi à vérifier la révision du document, reportez-vous à la section Erreurs courantes de vérification pour obtenir des informations sur les causes possibles.

Utiliser une preuve pour recalculer votre résumé

Une fois que QLDB a renvoyé une preuve pour votre demande de vérification de documents, vous pouvez essayer d'effectuer vous-même les calculs de hachage. Cette section décrit les étapes de haut niveau pour recalculer votre résumé à l'aide des preuves fournies.

Tout d'abord, associez votre hachage de révision au premier hachage de la liste des hachages de preuve. Procédez ensuite comme suit.

1. Triez les deux hachages. Comparez les hachages en fonction de leurs valeurs d'octets signées dans l'ordre little-endian.
2. Concaténez les deux hachages dans l'ordre trié.
3. Hachez la paire concaténée avec un générateur de hachage SHA-256.
4. Associez votre nouveau hachage au hachage suivant de la preuve et répétez les étapes 1 à 3. Une fois que vous avez traité le dernier hachage de preuve, votre nouveau hachage est votre condensé recalculé.

Si votre résumé recalculé correspond à votre résumé enregistré précédemment, votre document est vérifié avec succès.

Pour un step-by-step didacticiel contenant des exemples de code illustrant ces étapes de vérification, passez àTutoriel : vérification des données à l'aide d'un AWS SDK.

Tutoriel : vérification des données à l'aide d'un AWS SDK

Dans ce didacticiel, vous allez vérifier un hachage de révision de document et un hachage de bloc de journal dans un registre Amazon QLDB à l'aide de l'API QLDB via un SDK. AWS Vous utilisez également le pilote QLDB pour demander la révision du document.

Prenons l'exemple d'une révision de document contenant des données relatives à un véhicule dont le numéro d'identification du véhicule (VIN) est de`KM8SRDHF6EU074761`. La révision du document se trouve dans une `VehicleRegistration` table figurant dans un registre nommé`vehicle-registration`. Supposons que vous souhaitiez vérifier l'intégrité de la révision du document pour ce véhicule et du bloc de journal qui contient la révision.

> **Note**
>
> Pour un article de AWS blog détaillé qui aborde la valeur de la vérification cryptographique dans le contexte d'un cas d'utilisation réaliste, consultez la section Vérification cryptographique dans le monde réel avec Amazon QLDB.

Rubriques

Prérequis

Avant de commencer, assurez-vous d'effectuer les opérations suivantes :

1. Configurez le pilote QLDB pour la langue de votre choix en remplissant les conditions requises ci-dessous. Démarrage QLDB Cela inclut l'inscription AWS, l'octroi d'un accès programmatique pour le développement et la configuration de votre environnement de développement.
2. Suivez les étapes 1 et 2 ci-dessous Mise en route avec la console Amazon QLDB pour créer un registre nommé `vehicle-registration` et le charger avec des exemples de données prédéfinis.

Passez ensuite en revue les étapes suivantes pour savoir comment fonctionne la vérification, puis exécutez l'exemple de code complet du début à la fin.

Étape 1 : demander un résumé

Avant de pouvoir vérifier les données, vous devez d'abord demander un résumé de votre registre `vehicle-registration` pour une utilisation ultérieure.

Java

```
// Get a digest
GetDigestRequest digestRequest = new GetDigestRequest().withName(ledgerName);
GetDigestResult digestResult = client.getDigest(digestRequest);

java.nio.ByteBuffer digest = digestResult.getDigest();

// expectedDigest is the buffer we will use later to compare against our calculated
 digest
byte[] expectedDigest = new byte[digest.remaining()];
digest.get(expectedDigest);
```

.NET

```
// Get a digest
GetDigestRequest getDigestRequest = new GetDigestRequest
{
    Name = ledgerName
};
GetDigestResponse getDigestResponse =
 client.GetDigestAsync(getDigestRequest).Result;

// expectedDigest is the buffer we will use later to compare against our calculated
 digest
```

```
MemoryStream digest = getDigestResponse.Digest;
byte[] expectedDigest = digest.ToArray();
```

Go

```
// Get a digest
currentLedgerName := ledgerName
input := qldb.GetDigestInput{Name: &currentLedgerName}
digestOutput, err := client.GetDigest(&input)
if err != nil {
    panic(err)
}

// expectedDigest is the buffer we will later use to compare against our calculated
 digest
expectedDigest := digestOutput.Digest
```

Node.js

```
// Get a digest
const getDigestRequest: GetDigestRequest = {
    Name: ledgerName
};
const getDigestResponse: GetDigestResponse = await
 qldbClient.getDigest(getDigestRequest).promise();

// expectedDigest is the buffer we will later use to compare against our calculated
 digest
const expectedDigest: Uint8Array = <Uint8Array>getDigestResponse.Digest;
```

Python

```
# Get a digest
get_digest_response = qldb_client.get_digest(Name=ledger_name)

# expected_digest is the buffer we will later use to compare against our calculated
 digest
expected_digest = get_digest_response.get('Digest')
digest_tip_address = get_digest_response.get('DigestTipAddress')
```

Étape 2 : demander la révision du document

Utilisez le pilote QLDB pour interroger les adresses de bloc, les hachages et les identifiants de document associés au VIN. `KM8SRDHF6EU074761`

Java

```
// Retrieve info for the given vin's document revisions
Result result = driver.execute(txn -> {
    final String query = String.format("SELECT blockAddress, hash, metadata.id FROM
 _ql_committed_%s WHERE data.VIN = '%s'", tableName, vin);
    return txn.execute(query);
});
```

.NET

```
// Retrieve info for the given vin's document revisions
var result = driver.Execute(txn => {
    string query = $"SELECT blockAddress, hash, metadata.id FROM
 _ql_committed_{tableName} WHERE data.VIN = '{vin}'";
    return txn.Execute(query);
});
```

Go

```
// Retrieve info for the given vin's document revisions
result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
    statement := fmt.Sprintf(
            "SELECT blockAddress, hash, metadata.id FROM _ql_committed_%s WHERE
 data.VIN = '%s'",
            tableName,
            vin)
    result, err := txn.Execute(statement)
    if err != nil {
        return nil, err
    }

    results := make([]map[string]interface{}, 0)

    // Convert the result set into a map
    for result.Next(txn) {
```

```
        var doc map[string]interface{}
        err := ion.Unmarshal(result.GetCurrentData(), &doc)
        if err != nil {
            return nil, err
        }
        results = append(results, doc)
    }
    return results, nil
})
if err != nil {
    panic(err)
}
resultSlice := result.([]map[string]interface{})
```

Node.js

```
const result: dom.Value[] = await driver.executeLambda(async (txn:
 TransactionExecutor): Promise<dom.Value[]> => {
    const query: string = `SELECT blockAddress, hash, metadata.id FROM
 _ql_committed_${tableName} WHERE data.VIN = '${vin}'`;
    const queryResult: Result = await txn.execute(query);
    return queryResult.getResultList();
});
```

Python

```
def query_doc_revision(txn):
    query = "SELECT blockAddress, hash, metadata.id FROM _ql_committed_{} WHERE
 data.VIN = '{}'".format(table_name, vin)
    return txn.execute_statement(query)

# Retrieve info for the given vin's document revisions
result = qldb_driver.execute_lambda(query_doc_revision)
```

Étape 3 : demander une preuve pour la révision

Parcourez les résultats de la requête et utilisez chaque adresse de bloc et chaque identifiant de document ainsi que le nom du grand livre pour soumettre une `GetRevision` demande. Pour obtenir une preuve de la révision, vous devez également fournir l'adresse du pourboire figurant dans le résumé enregistré précédemment. Cette opération d'API renvoie un objet qui inclut la révision du document et la preuve de la révision.

Pour plus d'informations sur la structure de révision et son contenu, consultezInterroger les métadonnées d'un document.

Java

```
for (IonValue ionValue : result) {
    IonStruct ionStruct = (IonStruct)ionValue;

    // Get the requested fields
    IonValue blockAddress = ionStruct.get("blockAddress");
    IonBlob hash = (IonBlob)ionStruct.get("hash");
    String metadataId = ((IonString)ionStruct.get("id")).stringValue();

    System.out.printf("Verifying document revision for id '%s'%n", metadataId);

    String blockAddressText = blockAddress.toString();

    // Submit a request for the revision
    GetRevisionRequest revisionRequest = new GetRevisionRequest()
            .withName(ledgerName)
            .withBlockAddress(new ValueHolder().withIonText(blockAddressText))
            .withDocumentId(metadataId)
            .withDigestTipAddress(digestResult.getDigestTipAddress());

    // Get a result back
    GetRevisionResult revisionResult = client.getRevision(revisionRequest);

    ...
}
```

.NET

```
foreach (IIonValue ionValue in result)
{
    IIonStruct ionStruct = ionValue;

    // Get the requested fields
    IIonValue blockAddress = ionStruct.GetField("blockAddress");
    IIonBlob hash = ionStruct.GetField("hash");
    String metadataId = ionStruct.GetField("id").StringValue;

    Console.WriteLine($"Verifying document revision for id '{metadataId}'");
```

```
    // Use an Ion Reader to convert block address to text
    IIonReader reader = IonReaderBuilder.Build(blockAddress);
    StringWriter sw = new StringWriter();
    IIonWriter textWriter = IonTextWriterBuilder.Build(sw);
    textWriter.WriteValues(reader);
    string blockAddressText = sw.ToString();

    // Submit a request for the revision
    GetRevisionRequest revisionRequest = new GetRevisionRequest
    {
        Name = ledgerName,
        BlockAddress = new ValueHolder
        {
            IonText = blockAddressText
        },
        DocumentId = metadataId,
        DigestTipAddress = getDigestResponse.DigestTipAddress
    };

    // Get a response back
    GetRevisionResponse revisionResponse =
 client.GetRevisionAsync(revisionRequest).Result;

    ...
}
```

Go

```
for _, value := range resultSlice {
    // Get the requested fields
    ionBlockAddress, err := ion.MarshalText(value["blockAddress"])
    if err != nil {
        panic(err)
    }
    blockAddress := string(ionBlockAddress)
    metadataId := value["id"].(string)
    documentHash := value["hash"].([]byte)

    fmt.Printf("Verifying document revision for id '%s'\n", metadataId)

    // Submit a request for the revision
    revisionInput := qldb.GetRevisionInput{
        BlockAddress:     &qldb.ValueHolder{IonText: &blockAddress},
```

```
        DigestTipAddress: digestOutput.DigestTipAddress,
        DocumentId:       &metadataId,
        Name:             &currentLedgerName,
    }

    // Get a result back
    revisionOutput, err := client.GetRevision(&revisionInput)
    if err != nil {
        panic(err)
    }

    ...
}
```

Node.js

```
for (let value of result) {
    // Get the requested fields
    const blockAddress: dom.Value = value.get("blockAddress");
    const hash: dom.Value = value.get("hash");
    const metadataId: string = value.get("id").stringValue();

    console.log(`Verifying document revision for id '${metadataId}'`);

    // Submit a request for the revision
    const revisionRequest: GetRevisionRequest = {
        Name: ledgerName,
        BlockAddress: {
            IonText: dumpText(blockAddress)
        },
        DocumentId: metadataId,
        DigestTipAddress: getDigestResponse.DigestTipAddress
    };

    // Get a response back
    const revisionResponse: GetRevisionResponse = await
 qldbClient.getRevision(revisionRequest).promise();

    ...
}
```

Python

```
for value in result:
    # Get the requested fields
    block_address = value['blockAddress']
    document_hash = value['hash']
    metadata_id = value['id']

    print("Verifying document revision for id '{}'".format(metadata_id))

    # Submit a request for the revision and get a result back
    proof_response = qldb_client.get_revision(Name=ledger_name,

 BlockAddress=block_address_to_dictionary(block_address),
                                                DocumentId=metadata_id,
                                                DigestTipAddress=digest_tip_address)
```

Ensuite, récupérez la preuve de la révision demandée.

L'API QLDB renvoie la preuve sous forme de chaîne de la liste ordonnée des hachages de nœuds. Pour convertir cette chaîne en une liste de la représentation binaire des hachages de nœuds, vous pouvez utiliser un lecteur Ion de la bibliothèque Amazon Ion. Pour plus d'informations sur l'utilisation de la bibliothèque Ion, consultez le livre de recettes Amazon Ion.

Java

Dans cet exemple, vous utilisez `IonReader` pour effectuer la conversion binaire.

```
String proofText = revisionResult.getProof().getIonText();

// Take the proof and convert it to a list of byte arrays
List<byte[]> internalHashes = new ArrayList<>();
IonReader reader = SYSTEM.newReader(proofText);
reader.next();
reader.stepIn();
while (reader.next() != null) {
    internalHashes.add(reader.newBytes());
}
```

.NET

Dans cet exemple, vous l'utilisez `IonLoader` pour charger la preuve dans un datagramme ionique.

```
string proofText = revisionResponse.Proof.IonText;
IIonDatagram proofValue = IonLoader.Default.Load(proofText);
```

Go

Dans cet exemple, vous utilisez un lecteur Ion pour convertir la preuve en binaire et pour parcourir la liste des hachages de nœuds de la preuve.

```
proofText := revisionOutput.Proof.IonText

// Use ion.Reader to iterate over the proof's node hashes
reader := ion.NewReaderString(*proofText)
// Enter the struct containing node hashes
reader.Next()
if err := reader.StepIn(); err != nil {
    panic(err)
}
```

Node.js

Dans cet exemple, vous utilisez la `load` fonction pour effectuer la conversion binaire.

```
let proofValue: dom.Value = load(revisionResponse.Proof.IonText);
```

Python

Dans cet exemple, vous utilisez la `loads` fonction pour effectuer la conversion binaire.

```
proof_text = proof_response.get('Proof').get('IonText')
proof_hashes = loads(proof_text)
```

Étape 4 : Recalculer le résumé à partir de la révision

Utilisez la liste des hachages de la preuve pour recalculer le résumé, en commençant par le hachage de révision. Tant que le résumé précédemment enregistré est connu et fiable en dehors de QLDB,

l'intégrité de la révision du document est prouvée si le hachage du résumé recalculé correspond au hachage du résumé enregistré.

Java

```
// Calculate digest
byte[] calculatedDigest = internalHashes.stream().reduce(hash.getBytes(),
 BlockHashVerification::dot);

boolean verified = Arrays.equals(expectedDigest, calculatedDigest);

if (verified) {
    System.out.printf("Successfully verified document revision for id '%s'!%n",
 metadataId);
} else {
    System.out.printf("Document revision for id '%s' verification failed!%n",
 metadataId);
    return;
}
```

.NET

```
byte[] documentHash = hash.Bytes().ToArray();
foreach (IIonValue proofHash in proofValue.GetElementAt(0))
{
    // Calculate the digest
    documentHash = Dot(documentHash, proofHash.Bytes().ToArray());
}

bool verified = expectedDigest.SequenceEqual(documentHash);

if (verified)
{
    Console.WriteLine($"Successfully verified document revision for id
 '{metadataId}'!");
}
else
{
    Console.WriteLine($"Document revision for id '{metadataId}' verification
 failed!");
    return;
}
```

Go

```
// Going through nodes and calculate digest
for reader.Next() {
    val, _ := reader.ByteValue()
    documentHash, err = dot(documentHash, val)
}

// Compare documentHash with the expected digest
verified := reflect.DeepEqual(documentHash, expectedDigest)

if verified {
    fmt.Printf("Successfully verified document revision for id '%s'!\n", metadataId)
} else {
    fmt.Printf("Document revision for id '%s' verification failed!\n", metadataId)
    return
}
```

Node.js

```
let documentHash: Uint8Array = hash.uInt8ArrayValue();
proofValue.elements().forEach((proofHash: dom.Value) => {
    // Calculate the digest
    documentHash = dot(documentHash, proofHash.uInt8ArrayValue());
});

let verified: boolean = isEqual(expectedDigest, documentHash);

if (verified) {
    console.log(`Successfully verified document revision for id '${metadataId}'!`);
} else {
    console.log(`Document revision for id '${metadataId}' verification failed!`);
    return;
}
```

Python

```
# Calculate digest
calculated_digest = reduce(dot, proof_hashes, document_hash)

verified = calculated_digest == expected_digest
if verified:
```

```
    print("Successfully verified document revision for id
 '{}'!".format(metadata_id))
else:
    print("Document revision for id '{}' verification failed!".format(metadata_id))
```

Étape 5 : Demandez une preuve pour le bloc de journal

Ensuite, vous devez vérifier le bloc de journal qui contient la révision du document.

Utilisez l'adresse de blocage et l'adresse du pourboire figurant dans le résumé que vous avez enregistré à l'étape 1 pour soumettre une `GetBlock` demande. Comme pour la `GetRevision` demande de l'étape 2, vous devez à nouveau fournir l'adresse du pourboire figurant dans le résumé enregistré pour obtenir une preuve du blocage. Cette opération d'API renvoie un objet qui inclut le bloc et la preuve du bloc.

Pour plus d'informations sur la structure des blocs de journal et son contenu, consultezContenu du journal dans Amazon QLDB.

Java

```
// Submit a request for the block
GetBlockRequest getBlockRequest = new GetBlockRequest()
        .withName(ledgerName)
        .withBlockAddress(new ValueHolder().withIonText(blockAddressText))
        .withDigestTipAddress(digestResult.getDigestTipAddress());

// Get a result back
GetBlockResult getBlockResult = client.getBlock(getBlockRequest);
```

.NET

```
// Submit a request for the block
GetBlockRequest getBlockRequest = new GetBlockRequest
{
    Name = ledgerName,
    BlockAddress = new ValueHolder
    {
        IonText = blockAddressText
    },
    DigestTipAddress = getDigestResponse.DigestTipAddress
};
```

```
// Get a response back
GetBlockResponse getBlockResponse = client.GetBlockAsync(getBlockRequest).Result;
```

Go

```
// Submit a request for the block
blockInput := qldb.GetBlockInput{
    Name:             &currentLedgerName,
    BlockAddress:     &qldb.ValueHolder{IonText: &blockAddress},
    DigestTipAddress: digestOutput.DigestTipAddress,
}

// Get a result back
blockOutput, err := client.GetBlock(&blockInput)
if err != nil {
    panic(err)
}
```

Node.js

```
// Submit a request for the block
const getBlockRequest: GetBlockRequest = {
    Name: ledgerName,
    BlockAddress: {
        IonText: dumpText(blockAddress)
    },
    DigestTipAddress: getDigestResponse.DigestTipAddress
};

// Get a response back
const getBlockResponse: GetBlockResponse = await
 qldbClient.getBlock(getBlockRequest).promise();
```

Python

```
def block_address_to_dictionary(ion_dict):
"""
Convert a block address from IonPyDict into a dictionary.
Shape of the dictionary must be: {'IonText': "{strandId: <"strandId">, sequenceNo:
 <sequenceNo>}"}

:type ion_dict: :py:class:`amazon.ion.simple_types.IonPyDict`/str
```

```
:param ion_dict: The block address value to convert.

:rtype: dict
:return: The converted dict.
"""
block_address = {'IonText': {}}
if not isinstance(ion_dict, str):
    py_dict = '{{strandId: "{}", sequenceNo:{}}}'.format(ion_dict['strandId'],
 ion_dict['sequenceNo'])
    ion_dict = py_dict
block_address['IonText'] = ion_dict
return block_address

# Submit a request for the block and get a result back
block_response = qldb_client.get_block(Name=ledger_name,
 BlockAddress=block_address_to_dictionary(block_address),
                                        DigestTipAddress=digest_tip_address)
```

Ensuite, récupérez le hachage du bloc et la preuve à partir du résultat.

Java

Dans cet exemple, vous l'utilisez `IonLoader` pour charger l'objet du bloc dans un `IonDatagram` conteneur.

```
String blockText = getBlockResult.getBlock().getIonText();

IonDatagram datagram = SYSTEM.getLoader().load(blockText);
ionStruct = (IonStruct)datagram.get(0);

final byte[] blockHash = ((IonBlob)ionStruct.get("blockHash")).getBytes();
```

Vous l'utilisez également `IonLoader` pour charger la preuve dans un`IonDatagram`.

```
proofText = getBlockResult.getProof().getIonText();

// Take the proof and create a list of hash binary data
datagram = SYSTEM.getLoader().load(proofText);
ListIterator<IonValue> listIter =
 ((IonList)datagram.iterator().next()).listIterator();

internalHashes.clear();
```

```
while (listIter.hasNext()) {
    internalHashes.add(((IonBlob)listIter.next()).getBytes());
}
```

.NET

Dans cet exemple, vous pouvez `IonLoader` charger le bloc et la preuve dans un datagramme ionique pour chacun d'eux.

```
string blockText = getBlockResponse.Block.IonText;
IIonDatagram blockValue = IonLoader.Default.Load(blockText);

// blockValue is a IonDatagram, and the first value is an IonStruct containing the
 blockHash
byte[] blockHash =
 blockValue.GetElementAt(0).GetField("blockHash").Bytes().ToArray();

proofText = getBlockResponse.Proof.IonText;
proofValue = IonLoader.Default.Load(proofText);
```

Go

Dans cet exemple, vous utilisez un lecteur Ion pour convertir la preuve en binaire et pour parcourir la liste des hachages de nœuds de la preuve.

```
proofText = blockOutput.Proof.IonText

block := new(map[string]interface{})
err = ion.UnmarshalString(*blockOutput.Block.IonText, block)
if err != nil {
    panic(err)
}

blockHash := (*block)["blockHash"].([]byte)

// Use ion.Reader to iterate over the proof's node hashes
reader = ion.NewReaderString(*proofText)
// Enter the struct containing node hashes
reader.Next()
if err := reader.StepIn(); err != nil {
    panic(err)
}
```

Node.js

Dans cet exemple, vous utilisez la `load` fonction pour convertir le bloc et la preuve en binaire.

```
const blockValue: dom.Value = load(getBlockResponse.Block.IonText)
let blockHash: Uint8Array = blockValue.get("blockHash").uInt8ArrayValue();

proofValue = load(getBlockResponse.Proof.IonText);
```

Python

Dans cet exemple, vous utilisez la `loads` fonction pour convertir le bloc et la preuve en binaire.

```
block_text = block_response.get('Block').get('IonText')
block = loads(block_text)

block_hash = block.get('blockHash')

proof_text = block_response.get('Proof').get('IonText')
proof_hashes = loads(proof_text)
```

Étape 6 : Recalculer le condensé à partir du bloc

Utilisez la liste des hachages de la preuve pour recalculer le condensé, en commençant par le hachage du bloc. Tant que le condensé précédemment enregistré est connu et fiable en dehors de QLDB, l'intégrité du bloc est prouvée si le hachage du condensé recalculé correspond au hachage du condensé enregistré.

Java

```
// Calculate digest
calculatedDigest = internalHashes.stream().reduce(blockHash,
 BlockHashVerification::dot);

verified = Arrays.equals(expectedDigest, calculatedDigest);

if (verified) {
    System.out.printf("Block address '%s' successfully verified!%n",
 blockAddressText);
} else {
```

```
    System.out.printf("Block address '%s' verification failed!%n",
 blockAddressText);
}
```

.NET

```
foreach (IIonValue proofHash in proofValue.GetElementAt(0))
{
    // Calculate the digest
    blockHash = Dot(blockHash, proofHash.Bytes().ToArray());
}

verified = expectedDigest.SequenceEqual(blockHash);

if (verified)
{
    Console.WriteLine($"Block address '{blockAddressText}' successfully verified!");
}
else
{
    Console.WriteLine($"Block address '{blockAddressText}' verification failed!");
}
```

Go

```
// Going through nodes and calculate digest
for reader.Next() {
    val, err := reader.ByteValue()
    if err != nil {
        panic(err)
    }
    blockHash, err = dot(blockHash, val)
}

// Compare blockHash with the expected digest
verified = reflect.DeepEqual(blockHash, expectedDigest)

if verified {
    fmt.Printf("Block address '%s' successfully verified!\n", blockAddress)
} else {
    fmt.Printf("Block address '%s' verification failed!\n", blockAddress)
    return
}
```

Node.js

```
proofValue.elements().forEach((proofHash: dom.Value) => {
    // Calculate the digest
    blockHash = dot(blockHash, proofHash.uInt8ArrayValue());
});

verified = isEqual(expectedDigest, blockHash);

if (verified) {
    console.log(`Block address '${dumpText(blockAddress)}' successfully verified!`);
} else {
    console.log(`Block address '${dumpText(blockAddress)}' verification failed!`);
}
```

Python

```
# Calculate digest
calculated_digest = reduce(dot, proof_hashes, block_hash)

verified = calculated_digest == expected_digest
if verified:
    print("Block address '{}' successfully verified!".format(dumps(block_address,
                                                                   binary=False,

 omit_version_marker=True)))
else:
    print("Block address '{}' verification failed!".format(block_address))
```

Les exemples de code précédents utilisent la `dot` fonction suivante lors du recalcul du condensé. Cette fonction prend une entrée de deux hachages, les trie, les concatène, puis renvoie le hachage du tableau concaténé.

Java

```
/**
 * Takes two hashes, sorts them, concatenates them, and then returns the
 * hash of the concatenated array.
 *
 * @param h1
 *              Byte array containing one of the hashes to compare.
```

```
     * @param h2
     *              Byte array containing one of the hashes to compare.
     * @return the concatenated array of hashes.
     */
    public static byte[] dot(final byte[] h1, final byte[] h2) {
        if (h1.length != HASH_LENGTH || h2.length != HASH_LENGTH) {
            throw new IllegalArgumentException("Invalid hash.");
        }

        int byteEqual = 0;
        for (int i = h1.length - 1; i >= 0; i--) {
            byteEqual = Byte.compare(h1[i], h2[i]);
            if (byteEqual != 0) {
                break;
            }
        }

        byte[] concatenated = new byte[h1.length + h2.length];
        if (byteEqual < 0) {
            System.arraycopy(h1, 0, concatenated, 0, h1.length);
            System.arraycopy(h2, 0, concatenated, h1.length, h2.length);
        } else {
            System.arraycopy(h2, 0, concatenated, 0, h2.length);
            System.arraycopy(h1, 0, concatenated, h2.length, h1.length);
        }

        MessageDigest messageDigest;
        try {
            messageDigest = MessageDigest.getInstance("SHA-256");
        } catch (NoSuchAlgorithmException e) {
            throw new IllegalStateException("SHA-256 message digest is unavailable", e);
        }

        messageDigest.update(concatenated);
        return messageDigest.digest();
    }
```

.NET

```
/// <summary>
/// Takes two hashes, sorts them, concatenates them, and then returns the
/// hash of the concatenated array.
/// </summary>
```

```
/// <param name="h1">Byte array containing one of the hashes to compare.</param>
/// <param name="h2">Byte array containing one of the hashes to compare.</param>
/// <returns>The concatenated array of hashes.</returns>
private static byte[] Dot(byte[] h1, byte[] h2)
{
    if (h1.Length == 0)
    {
        return h2;
    }

    if (h2.Length == 0)
    {
        return h1;
    }

    HashAlgorithm hashAlgorithm = HashAlgorithm.Create("SHA256");
    HashComparer comparer = new HashComparer();
    if (comparer.Compare(h1, h2) < 0)
    {
        return hashAlgorithm.ComputeHash(h1.Concat(h2).ToArray());
    }
    else
    {
        return hashAlgorithm.ComputeHash(h2.Concat(h1).ToArray());
    }
}

private class HashComparer : IComparer<byte[]>
{
    private static readonly int HASH_LENGTH = 32;

    public int Compare(byte[] h1, byte[] h2)
    {
        if (h1.Length != HASH_LENGTH || h2.Length != HASH_LENGTH)
        {
            throw new ArgumentException("Invalid hash");
        }

        for (var i = h1.Length - 1; i >= 0; i--)
        {
            var byteEqual = (sbyte)h1[i] - (sbyte)h2[i];
            if (byteEqual != 0)
            {
                return byteEqual;
```

```
            }
        }

        return 0;
    }
}
```

Go

```
// Takes two hashes, sorts them, concatenates them, and then returns the hash of the
 concatenated array.
func dot(h1, h2 []byte) ([]byte, error) {
    compare, err := hashComparator(h1, h2)
    if err != nil {
        return nil, err
    }

    var concatenated []byte
    if compare < 0 {
        concatenated = append(h1, h2...)
    } else {
        concatenated = append(h2, h1...)
    }

    newHash := sha256.Sum256(concatenated)
    return newHash[:], nil
}

func hashComparator(h1 []byte, h2 []byte) (int16, error) {
    if len(h1) != hashLength || len(h2) != hashLength {
        return 0, errors.New("invalid hash")
    }
    for i := range h1 {
        // Reverse index for little endianness
        index := hashLength - 1 - i

        // Handle byte being unsigned and overflow
        h1Int := int16(h1[index])
        h2Int := int16(h2[index])
        if h1Int > 127 {
            h1Int = 0 - (256 - h1Int)
        }
        if h2Int > 127 {
```

```
            h2Int = 0 - (256 - h2Int)
        }

        difference := h1Int - h2Int
        if difference != 0 {
            return difference, nil
        }
    }
    return 0, nil
}
```

Node.js

```
/**
 * Takes two hashes, sorts them, concatenates them, and calculates a digest based on
 the concatenated hash.
 * @param h1 Byte array containing one of the hashes to compare.
 * @param h2 Byte array containing one of the hashes to compare.
 * @returns The digest calculated from the concatenated hash values.
 */
function dot(h1: Uint8Array, h2: Uint8Array): Uint8Array {
    if (h1.length === 0) {
        return h2;
    }
    if (h2.length === 0) {
        return h1;
    }

    const newHashLib = createHash("sha256");

    let concatenated: Uint8Array;
    if (hashComparator(h1, h2) < 0) {
        concatenated = concatenate(h1, h2);
    } else {
        concatenated = concatenate(h2, h1);
    }
    newHashLib.update(concatenated);
    return newHashLib.digest();
}

/**
 * Compares two hashes by their **signed** byte values in little-endian order.
 * @param hash1 The hash value to compare.
```

```
 * @param hash2 The hash value to compare.
 * @returns Zero if the hash values are equal, otherwise return the difference of
 the first pair of non-matching
 *          bytes.
 * @throws RangeError When the hash is not the correct hash size.
 */
function hashComparator(hash1: Uint8Array, hash2: Uint8Array): number {
    if (hash1.length !== HASH_SIZE || hash2.length !== HASH_SIZE) {
        throw new RangeError("Invalid hash.");
    }
    for (let i = hash1.length-1; i >= 0; i--) {
        const difference: number = (hash1[i]<<24 >>24) - (hash2[i]<<24 >>24);
        if (difference !== 0) {
            return difference;
        }
    }
    return 0;
}

/**
 * Helper method that concatenates two Uint8Array.
 * @param arrays List of arrays to concatenate, in the order provided.
 * @returns The concatenated array.
 */
function concatenate(...arrays: Uint8Array[]): Uint8Array {
    let totalLength = 0;
    for (const arr of arrays) {
        totalLength += arr.length;
    }
    const result = new Uint8Array(totalLength);
    let offset = 0;
    for (const arr of arrays) {
        result.set(arr, offset);
        offset += arr.length;
    }
    return result;
}

/**
 * Helper method that checks for equality between two Uint8Array.
 * @param expected Byte array containing one of the hashes to compare.
 * @param actual Byte array containing one of the hashes to compare.
 * @returns Boolean indicating equality between the two Uint8Array.
 */
```

```
function isEqual(expected: Uint8Array, actual: Uint8Array): boolean {
    if (expected === actual) return true;
    if (expected == null || actual == null) return false;
    if (expected.length !== actual.length) return false;

    for (let i = 0; i < expected.length; i++) {
        if (expected[i] !== actual[i]) {
            return false;
        }
    }
    return true;
}
```

Python

```
def dot(hash1, hash2):
    """
    Takes two hashes, sorts them, concatenates them, and then returns the
    hash of the concatenated array.

    :type hash1: bytes
    :param hash1: The hash value to compare.

    :type hash2: bytes
    :param hash2: The hash value to compare.

    :rtype: bytes
    :return: The new hash value generated from concatenated hash values.
    """
    if len(hash1) != hash_length or len(hash2) != hash_length:
        raise ValueError('Illegal hash.')

    hash_array1 = array('b', hash1)
    hash_array2 = array('b', hash2)

    difference = 0
    for i in range(len(hash_array1) - 1, -1, -1):
        difference = hash_array1[i] - hash_array2[i]
        if difference != 0:
            break

    if difference < 0:
        concatenated = hash1 + hash2
```

```
    else:
        concatenated = hash2 + hash1

    new_hash_lib = sha256()
    new_hash_lib.update(concatenated)
    new_digest = new_hash_lib.digest()
    return new_digest
```

Exécutez l'exemple de code complet

Exécutez l'exemple de code complet comme suit pour effectuer toutes les étapes précédentes du début à la fin.

Java

```
import com.amazon.ion.IonBlob;
import com.amazon.ion.IonDatagram;
import com.amazon.ion.IonList;
import com.amazon.ion.IonReader;
import com.amazon.ion.IonString;
import com.amazon.ion.IonStruct;
import com.amazon.ion.IonSystem;
import com.amazon.ion.IonValue;
import com.amazon.ion.system.IonSystemBuilder;
import com.amazonaws.services.qldb.AmazonQLDB;
import com.amazonaws.services.qldb.AmazonQLDBClientBuilder;
import com.amazonaws.services.qldb.model.GetBlockRequest;
import com.amazonaws.services.qldb.model.GetBlockResult;
import com.amazonaws.services.qldb.model.GetDigestRequest;
import com.amazonaws.services.qldb.model.GetDigestResult;
import com.amazonaws.services.qldb.model.GetRevisionRequest;
import com.amazonaws.services.qldb.model.GetRevisionResult;
import com.amazonaws.services.qldb.model.ValueHolder;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.ListIterator;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.qldbsession.QldbSessionClient;
```

```
import software.amazon.awssdk.services.qldbsession.QldbSessionClientBuilder;
import software.amazon.qldb.QldbDriver;
import software.amazon.qldb.Result;

public class BlockHashVerification {
    private static final IonSystem SYSTEM = IonSystemBuilder.standard().build();
    private static final QldbDriver driver = createQldbDriver();
    private static final AmazonQLDB client =
 AmazonQLDBClientBuilder.standard().build();
    private static final String region = "us-east-1";
    private static final String ledgerName = "vehicle-registration";
    private static final String tableName = "VehicleRegistration";
    private static final String vin = "KM8SRDHF6EU074761";
    private static final int HASH_LENGTH = 32;

    /**
     * Create a pooled driver for creating sessions.
     *
     * @return The pooled driver for creating sessions.
     */
    public static QldbDriver createQldbDriver() {
        QldbSessionClientBuilder sessionClientBuilder = QldbSessionClient.builder();
        sessionClientBuilder.region(Region.of(region));

        return QldbDriver.builder()
                .ledger(ledgerName)
                .sessionClientBuilder(sessionClientBuilder)
                .build();
    }

    /**
     * Takes two hashes, sorts them, concatenates them, and then returns the
     * hash of the concatenated array.
     *
     * @param h1
     *              Byte array containing one of the hashes to compare.
     * @param h2
     *              Byte array containing one of the hashes to compare.
     * @return the concatenated array of hashes.
     */
    public static byte[] dot(final byte[] h1, final byte[] h2) {
        if (h1.length != HASH_LENGTH || h2.length != HASH_LENGTH) {
            throw new IllegalArgumentException("Invalid hash.");
        }
```

```
        int byteEqual = 0;
        for (int i = h1.length - 1; i >= 0; i--) {
            byteEqual = Byte.compare(h1[i], h2[i]);
            if (byteEqual != 0) {
                break;
            }
        }

        byte[] concatenated = new byte[h1.length + h2.length];
        if (byteEqual < 0) {
            System.arraycopy(h1, 0, concatenated, 0, h1.length);
            System.arraycopy(h2, 0, concatenated, h1.length, h2.length);
        } else {
            System.arraycopy(h2, 0, concatenated, 0, h2.length);
            System.arraycopy(h1, 0, concatenated, h2.length, h1.length);
        }

        MessageDigest messageDigest;
        try {
            messageDigest = MessageDigest.getInstance("SHA-256");
        } catch (NoSuchAlgorithmException e) {
            throw new IllegalStateException("SHA-256 message digest is unavailable",
e);
        }

        messageDigest.update(concatenated);
        return messageDigest.digest();
    }

    public static void main(String[] args) {
        // Get a digest
        GetDigestRequest digestRequest = new
GetDigestRequest().withName(ledgerName);
        GetDigestResult digestResult = client.getDigest(digestRequest);

        java.nio.ByteBuffer digest = digestResult.getDigest();

        // expectedDigest is the buffer we will use later to compare against our
calculated digest
        byte[] expectedDigest = new byte[digest.remaining()];
        digest.get(expectedDigest);

        // Retrieve info for the given vin's document revisions
```

```
        Result result = driver.execute(txn -> {
            final String query = String.format("SELECT blockAddress, hash,
metadata.id FROM _ql_committed_%s WHERE data.VIN = '%s'", tableName, vin);
            return txn.execute(query);
        });

        System.out.printf("Verifying document revisions for vin '%s' in table '%s'
in ledger '%s'%n", vin, tableName, ledgerName);

        for (IonValue ionValue : result) {
            IonStruct ionStruct = (IonStruct)ionValue;

            // Get the requested fields
            IonValue blockAddress = ionStruct.get("blockAddress");
            IonBlob hash = (IonBlob)ionStruct.get("hash");
            String metadataId = ((IonString)ionStruct.get("id")).stringValue();

            System.out.printf("Verifying document revision for id '%s'%n",
metadataId);

            String blockAddressText = blockAddress.toString();

            // Submit a request for the revision
            GetRevisionRequest revisionRequest = new GetRevisionRequest()
                    .withName(ledgerName)
                    .withBlockAddress(new
ValueHolder().withIonText(blockAddressText))
                    .withDocumentId(metadataId)
                    .withDigestTipAddress(digestResult.getDigestTipAddress());

            // Get a result back
            GetRevisionResult revisionResult = client.getRevision(revisionRequest);

            String proofText = revisionResult.getProof().getIonText();

            // Take the proof and convert it to a list of byte arrays
            List<byte[]> internalHashes = new ArrayList<>();
            IonReader reader = SYSTEM.newReader(proofText);
            reader.next();
            reader.stepIn();
            while (reader.next() != null) {
                internalHashes.add(reader.newBytes());
            }
```

```
            // Calculate digest
            byte[] calculatedDigest =
internalHashes.stream().reduce(hash.getBytes(), BlockHashVerification::dot);

            boolean verified = Arrays.equals(expectedDigest, calculatedDigest);

            if (verified) {
                System.out.printf("Successfully verified document revision for id
'%s'!%n", metadataId);
            } else {
                System.out.printf("Document revision for id '%s' verification
failed!%n", metadataId);
                return;
            }

            // Submit a request for the block
            GetBlockRequest getBlockRequest = new GetBlockRequest()
                    .withName(ledgerName)
                    .withBlockAddress(new
ValueHolder().withIonText(blockAddressText))
                    .withDigestTipAddress(digestResult.getDigestTipAddress());

            // Get a result back
            GetBlockResult getBlockResult = client.getBlock(getBlockRequest);

            String blockText = getBlockResult.getBlock().getIonText();

            IonDatagram datagram = SYSTEM.getLoader().load(blockText);
            ionStruct = (IonStruct)datagram.get(0);

            final byte[] blockHash =
((IonBlob)ionStruct.get("blockHash")).getBytes();

            proofText = getBlockResult.getProof().getIonText();

            // Take the proof and create a list of hash binary data
            datagram = SYSTEM.getLoader().load(proofText);
            ListIterator<IonValue> listIter =
((IonList)datagram.iterator().next()).listIterator();

            internalHashes.clear();
            while (listIter.hasNext()) {
                internalHashes.add(((IonBlob)listIter.next()).getBytes());
            }
```

```
            // Calculate digest
            calculatedDigest = internalHashes.stream().reduce(blockHash,
 BlockHashVerification::dot);

            verified = Arrays.equals(expectedDigest, calculatedDigest);

            if (verified) {
                System.out.printf("Block address '%s' successfully verified!%n",
 blockAddressText);
            } else {
                System.out.printf("Block address '%s' verification failed!%n",
 blockAddressText);
            }
        }
    }
}
```

.NET

```
using Amazon.IonDotnet;
using Amazon.IonDotnet.Builders;
using Amazon.IonDotnet.Tree;
using Amazon.QLDB;
using Amazon.QLDB.Driver;
using Amazon.QLDB.Model;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Security.Cryptography;

namespace BlockHashVerification
{
    class BlockHashVerification
    {
        private static readonly string ledgerName = "vehicle-registration";
        private static readonly string tableName = "VehicleRegistration";
        private static readonly string vin = "KM8SRDHF6EU074761";
        private static readonly IQldbDriver driver =
 QldbDriver.Builder().WithLedger(ledgerName).Build();
        private static readonly IAmazonQLDB client = new AmazonQLDBClient();
```

```
        /// <summary>
        /// Takes two hashes, sorts them, concatenates them, and then returns the
        /// hash of the concatenated array.
        /// </summary>
        /// <param name="h1">Byte array containing one of the hashes to compare.</
param>
        /// <param name="h2">Byte array containing one of the hashes to compare.</
param>
        /// <returns>The concatenated array of hashes.</returns>
        private static byte[] Dot(byte[] h1, byte[] h2)
        {
            if (h1.Length == 0)
            {
                return h2;
            }

            if (h2.Length == 0)
            {
                return h1;
            }

            HashAlgorithm hashAlgorithm = HashAlgorithm.Create("SHA256");
            HashComparer comparer = new HashComparer();
            if (comparer.Compare(h1, h2) < 0)
            {
                return hashAlgorithm.ComputeHash(h1.Concat(h2).ToArray());
            }
            else
            {
                return hashAlgorithm.ComputeHash(h2.Concat(h1).ToArray());
            }
        }

        private class HashComparer : IComparer<byte[]>
        {
            private static readonly int HASH_LENGTH = 32;

            public int Compare(byte[] h1, byte[] h2)
            {
                if (h1.Length != HASH_LENGTH || h2.Length != HASH_LENGTH)
                {
                    throw new ArgumentException("Invalid hash");
                }
```

```
                for (var i = h1.Length - 1; i >= 0; i--)
                {
                    var byteEqual = (sbyte)h1[i] - (sbyte)h2[i];
                    if (byteEqual != 0)
                    {
                        return byteEqual;
                    }
                }

                return 0;
            }
        }

        static void Main()
        {
            // Get a digest
            GetDigestRequest getDigestRequest = new GetDigestRequest
            {
                Name = ledgerName
            };
            GetDigestResponse getDigestResponse =
client.GetDigestAsync(getDigestRequest).Result;

            // expectedDigest is the buffer we will use later to compare against our
calculated digest
            MemoryStream digest = getDigestResponse.Digest;
            byte[] expectedDigest = digest.ToArray();

            // Retrieve info for the given vin's document revisions
            var result = driver.Execute(txn => {
                string query = $"SELECT blockAddress, hash, metadata.id FROM
_ql_committed_{tableName} WHERE data.VIN = '{vin}'";
                return txn.Execute(query);
            });

            Console.WriteLine($"Verifying document revisions for vin '{vin}' in
table '{tableName}' in ledger '{ledgerName}'");

            foreach (IIonValue ionValue in result)
            {
                IIonStruct ionStruct = ionValue;

                // Get the requested fields
                IIonValue blockAddress = ionStruct.GetField("blockAddress");
```

```
                IIonBlob hash = ionStruct.GetField("hash");
                String metadataId = ionStruct.GetField("id").StringValue;

                Console.WriteLine($"Verifying document revision for id
'{metadataId}'");

                // Use an Ion Reader to convert block address to text
                IIonReader reader = IonReaderBuilder.Build(blockAddress);
                StringWriter sw = new StringWriter();
                IIonWriter textWriter = IonTextWriterBuilder.Build(sw);
                textWriter.WriteValues(reader);
                string blockAddressText = sw.ToString();

                // Submit a request for the revision
                GetRevisionRequest revisionRequest = new GetRevisionRequest
                {
                    Name = ledgerName,
                    BlockAddress = new ValueHolder
                    {
                        IonText = blockAddressText
                    },
                    DocumentId = metadataId,
                    DigestTipAddress = getDigestResponse.DigestTipAddress
                };

                // Get a response back
                GetRevisionResponse revisionResponse =
client.GetRevisionAsync(revisionRequest).Result;

                string proofText = revisionResponse.Proof.IonText;
                IIonDatagram proofValue = IonLoader.Default.Load(proofText);

                byte[] documentHash = hash.Bytes().ToArray();
                foreach (IIonValue proofHash in proofValue.GetElementAt(0))
                {
                    // Calculate the digest
                    documentHash = Dot(documentHash, proofHash.Bytes().ToArray());
                }

                bool verified = expectedDigest.SequenceEqual(documentHash);

                if (verified)
                {
```

```
                    Console.WriteLine($"Successfully verified document revision for
id '{metadataId}'!");
                }
                else
                {
                    Console.WriteLine($"Document revision for id '{metadataId}'
verification failed!");
                    return;
                }

                // Submit a request for the block
                GetBlockRequest getBlockRequest = new GetBlockRequest
                {
                    Name = ledgerName,
                    BlockAddress = new ValueHolder
                    {
                        IonText = blockAddressText
                    },
                    DigestTipAddress = getDigestResponse.DigestTipAddress
                };

                // Get a response back
                GetBlockResponse getBlockResponse =
client.GetBlockAsync(getBlockRequest).Result;

                string blockText = getBlockResponse.Block.IonText;
                IIonDatagram blockValue = IonLoader.Default.Load(blockText);

                // blockValue is a IonDatagram, and the first value is an IonStruct
containing the blockHash
                byte[] blockHash =
blockValue.GetElementAt(0).GetField("blockHash").Bytes().ToArray();

                proofText = getBlockResponse.Proof.IonText;
                proofValue = IonLoader.Default.Load(proofText);

                foreach (IIonValue proofHash in proofValue.GetElementAt(0))
                {
                    // Calculate the digest
                    blockHash = Dot(blockHash, proofHash.Bytes().ToArray());
                }

                verified = expectedDigest.SequenceEqual(blockHash);
```

```
                if (verified)
                {
                    Console.WriteLine($"Block address '{blockAddressText}'
 successfully verified!");
                }
                else
                {
                    Console.WriteLine($"Block address '{blockAddressText}'
 verification failed!");
                }
            }
        }
    }
}
```

Go

```
package main

import (
    "context"
    "crypto/sha256"
    "errors"
    "fmt"
    "reflect"

    "github.com/amzn/ion-go/ion"
    "github.com/aws/aws-sdk-go/aws"
    AWSSession "github.com/aws/aws-sdk-go/aws/session"
    "github.com/aws/aws-sdk-go/service/qldb"
    "github.com/aws/aws-sdk-go/service/qldbsession"
    "github.com/awslabs/amazon-qldb-driver-go/qldbdriver"
)

const (
    hashLength = 32
    ledgerName = "vehicle-registration"
    tableName  = "VehicleRegistration"
    vin        = "KM8SRDHF6EU074761"
)

// Takes two hashes, sorts them, concatenates them, and then returns the hash of the
 concatenated array.
```

```
func dot(h1, h2 []byte) ([]byte, error) {
    compare, err := hashComparator(h1, h2)
    if err != nil {
        return nil, err
    }

    var concatenated []byte
    if compare < 0 {
        concatenated = append(h1, h2...)
    } else {
        concatenated = append(h2, h1...)
    }

    newHash := sha256.Sum256(concatenated)
    return newHash[:], nil
}

func hashComparator(h1 []byte, h2 []byte) (int16, error) {
    if len(h1) != hashLength || len(h2) != hashLength {
        return 0, errors.New("invalid hash")
    }
    for i := range h1 {
        // Reverse index for little endianness
        index := hashLength - 1 - i

        // Handle byte being unsigned and overflow
        h1Int := int16(h1[index])
        h2Int := int16(h2[index])
        if h1Int > 127 {
            h1Int = 0 - (256 - h1Int)
        }
        if h2Int > 127 {
            h2Int = 0 - (256 - h2Int)
        }

        difference := h1Int - h2Int
        if difference != 0 {
            return difference, nil
        }
    }
    return 0, nil
}

func main() {
```

```
   driverSession := AWSSession.Must(AWSSession.NewSession(aws.NewConfig()))
   qldbSession := qldbsession.New(driverSession)
   driver, err := qldbdriver.New(ledgerName, qldbSession, func(options
*qldbdriver.DriverOptions) {})
   if err != nil {
       panic(err)
   }
   client := qldb.New(driverSession)

   // Get a digest
   currentLedgerName := ledgerName
   input := qldb.GetDigestInput{Name: &currentLedgerName}
   digestOutput, err := client.GetDigest(&input)
   if err != nil {
       panic(err)
   }

   // expectedDigest is the buffer we will later use to compare against our
calculated digest
   expectedDigest := digestOutput.Digest

   // Retrieve info for the given vin's document revisions
   result, err := driver.Execute(context.Background(), func(txn
qldbdriver.Transaction) (interface{}, error) {
       statement := fmt.Sprintf(
               "SELECT blockAddress, hash, metadata.id FROM _ql_committed_%s WHERE
data.VIN = '%s'",
               tableName,
               vin)
       result, err := txn.Execute(statement)
       if err != nil {
           return nil, err
       }

       results := make([]map[string]interface{}, 0)

       // Convert the result set into a map
       for result.Next(txn) {
           var doc map[string]interface{}
           err := ion.Unmarshal(result.GetCurrentData(), &doc)
           if err != nil {
               return nil, err
           }
           results = append(results, doc)
```

```
        }
        return results, nil
    })
    if err != nil {
        panic(err)
    }
    resultSlice := result.([]map[string]interface{})

    fmt.Printf("Verifying document revisions for vin '%s' in table '%s' in ledger
'%s'\n", vin, tableName, ledgerName)

    for _, value := range resultSlice {
        // Get the requested fields
        ionBlockAddress, err := ion.MarshalText(value["blockAddress"])
        if err != nil {
            panic(err)
        }
        blockAddress := string(ionBlockAddress)
        metadataId := value["id"].(string)
        documentHash := value["hash"].([]byte)

        fmt.Printf("Verifying document revision for id '%s'\n", metadataId)

        // Submit a request for the revision
        revisionInput := qldb.GetRevisionInput{
            BlockAddress:     &qldb.ValueHolder{IonText: &blockAddress},
            DigestTipAddress: digestOutput.DigestTipAddress,
            DocumentId:       &metadataId,
            Name:             &currentLedgerName,
        }

        // Get a result back
        revisionOutput, err := client.GetRevision(&revisionInput)
        if err != nil {
            panic(err)
        }

        proofText := revisionOutput.Proof.IonText

        // Use ion.Reader to iterate over the proof's node hashes
        reader := ion.NewReaderString(*proofText)
        // Enter the struct containing node hashes
        reader.Next()
        if err := reader.StepIn(); err != nil {
```

```
            panic(err)
        }

        // Going through nodes and calculate digest
        for reader.Next() {
            val, _ := reader.ByteValue()
            documentHash, err = dot(documentHash, val)
        }

        // Compare documentHash with the expected digest
        verified := reflect.DeepEqual(documentHash, expectedDigest)

        if verified {
            fmt.Printf("Successfully verified document revision for id '%s'!\n",
metadataId)
        } else {
            fmt.Printf("Document revision for id '%s' verification failed!\n",
metadataId)
            return
        }

        // Submit a request for the block
        blockInput := qldb.GetBlockInput{
            Name:             &currentLedgerName,
            BlockAddress:     &qldb.ValueHolder{IonText: &blockAddress},
            DigestTipAddress: digestOutput.DigestTipAddress,
        }

        // Get a result back
        blockOutput, err := client.GetBlock(&blockInput)
        if err != nil {
            panic(err)
        }

        proofText = blockOutput.Proof.IonText

        block := new(map[string]interface{})
        err = ion.UnmarshalString(*blockOutput.Block.IonText, block)
        if err != nil {
            panic(err)
        }

        blockHash := (*block)["blockHash"].([]byte)
```

```
        // Use ion.Reader to iterate over the proof's node hashes
        reader = ion.NewReaderString(*proofText)
        // Enter the struct containing node hashes
        reader.Next()
        if err := reader.StepIn(); err != nil {
            panic(err)
        }

        // Going through nodes and calculate digest
        for reader.Next() {
            val, err := reader.ByteValue()
            if err != nil {
                panic(err)
            }
            blockHash, err = dot(blockHash, val)
        }

        // Compare blockHash with the expected digest
        verified = reflect.DeepEqual(blockHash, expectedDigest)

        if verified {
            fmt.Printf("Block address '%s' successfully verified!\n", blockAddress)
        } else {
            fmt.Printf("Block address '%s' verification failed!\n", blockAddress)
            return
        }
    }
}
```

Node.js

```
import { QldbDriver, Result, TransactionExecutor} from "amazon-qldb-driver-nodejs";
import { QLDB } from "aws-sdk"
import { GetBlockRequest, GetBlockResponse, GetDigestRequest, GetDigestResponse,
 GetRevisionRequest, GetRevisionResponse } from "aws-sdk/clients/qldb";
import { createHash } from "crypto";
import { dom, dumpText, load } from "ion-js"

const ledgerName: string = "vehicle-registration";
const tableName: string = "VehicleRegistration";
const vin: string = "KM8SRDHF6EU074761";
const driver: QldbDriver = new QldbDriver(ledgerName);
const qldbClient: QLDB = new QLDB();
```

```
const HASH_SIZE = 32;

/**
 * Takes two hashes, sorts them, concatenates them, and calculates a digest based on
 the concatenated hash.
 * @param h1 Byte array containing one of the hashes to compare.
 * @param h2 Byte array containing one of the hashes to compare.
 * @returns The digest calculated from the concatenated hash values.
 */
function dot(h1: Uint8Array, h2: Uint8Array): Uint8Array {
    if (h1.length === 0) {
        return h2;
    }
    if (h2.length === 0) {
        return h1;
    }

    const newHashLib = createHash("sha256");

    let concatenated: Uint8Array;
    if (hashComparator(h1, h2) < 0) {
        concatenated = concatenate(h1, h2);
    } else {
        concatenated = concatenate(h2, h1);
    }
    newHashLib.update(concatenated);
    return newHashLib.digest();
}

/**
 * Compares two hashes by their **signed** byte values in little-endian order.
 * @param hash1 The hash value to compare.
 * @param hash2 The hash value to compare.
 * @returns Zero if the hash values are equal, otherwise return the difference of
 the first pair of non-matching
 *          bytes.
 * @throws RangeError When the hash is not the correct hash size.
 */
function hashComparator(hash1: Uint8Array, hash2: Uint8Array): number {
    if (hash1.length !== HASH_SIZE || hash2.length !== HASH_SIZE) {
        throw new RangeError("Invalid hash.");
    }
    for (let i = hash1.length-1; i >= 0; i--) {
        const difference: number = (hash1[i]<<24 >>24) - (hash2[i]<<24 >>24);
```

```
        if (difference !== 0) {
            return difference;
        }
    }
    return 0;
}

/**
 * Helper method that concatenates two Uint8Array.
 * @param arrays List of arrays to concatenate, in the order provided.
 * @returns The concatenated array.
 */
function concatenate(...arrays: Uint8Array[]): Uint8Array {
    let totalLength = 0;
    for (const arr of arrays) {
        totalLength += arr.length;
    }
    const result = new Uint8Array(totalLength);
    let offset = 0;
    for (const arr of arrays) {
        result.set(arr, offset);
        offset += arr.length;
    }
    return result;
}

/**
 * Helper method that checks for equality between two Uint8Array.
 * @param expected Byte array containing one of the hashes to compare.
 * @param actual Byte array containing one of the hashes to compare.
 * @returns Boolean indicating equality between the two Uint8Array.
 */
function isEqual(expected: Uint8Array, actual: Uint8Array): boolean {
    if (expected === actual) return true;
    if (expected == null || actual == null) return false;
    if (expected.length !== actual.length) return false;

    for (let i = 0; i < expected.length; i++) {
        if (expected[i] !== actual[i]) {
            return false;
        }
    }
    return true;
}
```

```
const main = async function (): Promise<void> {
    // Get a digest
    const getDigestRequest: GetDigestRequest = {
        Name: ledgerName
    };
    const getDigestResponse: GetDigestResponse = await
 qldbClient.getDigest(getDigestRequest).promise();

    // expectedDigest is the buffer we will later use to compare against our
 calculated digest
    const expectedDigest: Uint8Array = <Uint8Array>getDigestResponse.Digest;

    const result: dom.Value[] = await driver.executeLambda(async (txn:
 TransactionExecutor): Promise<dom.Value[]> => {
        const query: string = `SELECT blockAddress, hash, metadata.id FROM
 _ql_committed_${tableName} WHERE data.VIN = '${vin}'`;
        const queryResult: Result = await txn.execute(query);
        return queryResult.getResultList();
    });

    console.log(`Verifying document revisions for vin '${vin}' in table
 '${tableName}' in ledger '${ledgerName}'`);

    for (let value of result) {
        // Get the requested fields
        const blockAddress: dom.Value = value.get("blockAddress");
        const hash: dom.Value = value.get("hash");
        const metadataId: string = value.get("id").stringValue();

        console.log(`Verifying document revision for id '${metadataId}'`);

        // Submit a request for the revision
        const revisionRequest: GetRevisionRequest = {
            Name: ledgerName,
            BlockAddress: {
                IonText: dumpText(blockAddress)
            },
            DocumentId: metadataId,
            DigestTipAddress: getDigestResponse.DigestTipAddress
        };

        // Get a response back
```

```
        const revisionResponse: GetRevisionResponse = await
qldbClient.getRevision(revisionRequest).promise();

        let proofValue: dom.Value = load(revisionResponse.Proof.IonText);

        let documentHash: Uint8Array = hash.uInt8ArrayValue();
        proofValue.elements().forEach((proofHash: dom.Value) => {
            // Calculate the digest
            documentHash = dot(documentHash, proofHash.uInt8ArrayValue());
        });

        let verified: boolean = isEqual(expectedDigest, documentHash);

        if (verified) {
            console.log(`Successfully verified document revision for id
'${metadataId}'!`);
        } else {
            console.log(`Document revision for id '${metadataId}' verification
failed!`);
            return;
        }

        // Submit a request for the block
        const getBlockRequest: GetBlockRequest = {
            Name: ledgerName,
            BlockAddress: {
                IonText: dumpText(blockAddress)
            },
            DigestTipAddress: getDigestResponse.DigestTipAddress
        };

        // Get a response back
        const getBlockResponse: GetBlockResponse = await
qldbClient.getBlock(getBlockRequest).promise();

        const blockValue: dom.Value = load(getBlockResponse.Block.IonText)
        let blockHash: Uint8Array = blockValue.get("blockHash").uInt8ArrayValue();

        proofValue = load(getBlockResponse.Proof.IonText);

        proofValue.elements().forEach((proofHash: dom.Value) => {
            // Calculate the digest
            blockHash = dot(blockHash, proofHash.uInt8ArrayValue());
        });
```

```
        verified = isEqual(expectedDigest, blockHash);

        if (verified) {
            console.log(`Block address '${dumpText(blockAddress)}' successfully
 verified!`);
        } else {
            console.log(`Block address '${dumpText(blockAddress)}' verification
 failed!`);
        }
    }
};

if (require.main === module) {
    main();
}
```

Python

```
from amazon.ion.simpleion import dumps, loads
from array import array
from boto3 import client
from functools import reduce
from hashlib import sha256
from pyqldb.driver.qldb_driver import QldbDriver

ledger_name = 'vehicle-registration'
table_name = 'VehicleRegistration'
vin = 'KM8SRDHF6EU074761'
qldb_client = client('qldb')
hash_length = 32


def query_doc_revision(txn):
    query = "SELECT blockAddress, hash, metadata.id FROM _ql_committed_{} WHERE
 data.VIN = '{}'".format(table_name, vin)
    return txn.execute_statement(query)


def block_address_to_dictionary(ion_dict):
    """
    Convert a block address from IonPyDict into a dictionary.
```

```
    Shape of the dictionary must be: {'IonText': "{strandId: <"strandId">,
 sequenceNo: <sequenceNo>}"}

    :type ion_dict: :py:class:`amazon.ion.simple_types.IonPyDict`/str
    :param ion_dict: The block address value to convert.

    :rtype: dict
    :return: The converted dict.
    """
    block_address = {'IonText': {}}
    if not isinstance(ion_dict, str):
        py_dict = '{{strandId: "{}", sequenceNo:{}}}'.format(ion_dict['strandId'],
 ion_dict['sequenceNo'])
        ion_dict = py_dict
    block_address['IonText'] = ion_dict
    return block_address


def dot(hash1, hash2):
    """
    Takes two hashes, sorts them, concatenates them, and then returns the
    hash of the concatenated array.

    :type hash1: bytes
    :param hash1: The hash value to compare.

    :type hash2: bytes
    :param hash2: The hash value to compare.

    :rtype: bytes
    :return: The new hash value generated from concatenated hash values.
    """
    if len(hash1) != hash_length or len(hash2) != hash_length:
        raise ValueError('Illegal hash.')

    hash_array1 = array('b', hash1)
    hash_array2 = array('b', hash2)

    difference = 0
    for i in range(len(hash_array1) - 1, -1, -1):
        difference = hash_array1[i] - hash_array2[i]
        if difference != 0:
            break
```

```
    if difference < 0:
        concatenated = hash1 + hash2
    else:
        concatenated = hash2 + hash1

    new_hash_lib = sha256()
    new_hash_lib.update(concatenated)
    new_digest = new_hash_lib.digest()
    return new_digest


# Get a digest
get_digest_response = qldb_client.get_digest(Name=ledger_name)

# expected_digest is the buffer we will later use to compare against our calculated
 digest
expected_digest = get_digest_response.get('Digest')
digest_tip_address = get_digest_response.get('DigestTipAddress')

qldb_driver = QldbDriver(ledger_name=ledger_name)

# Retrieve info for the given vin's document revisions
result = qldb_driver.execute_lambda(query_doc_revision)

print("Verifying document revisions for vin '{}' in table '{}' in ledger
 '{}'".format(vin, table_name, ledger_name))

for value in result:
    # Get the requested fields
    block_address = value['blockAddress']
    document_hash = value['hash']
    metadata_id = value['id']

    print("Verifying document revision for id '{}'".format(metadata_id))

    # Submit a request for the revision and get a result back
    proof_response = qldb_client.get_revision(Name=ledger_name,

 BlockAddress=block_address_to_dictionary(block_address),
                                           DocumentId=metadata_id,
                                           DigestTipAddress=digest_tip_address)

    proof_text = proof_response.get('Proof').get('IonText')
    proof_hashes = loads(proof_text)
```

```
   # Calculate digest
   calculated_digest = reduce(dot, proof_hashes, document_hash)

   verified = calculated_digest == expected_digest
   if verified:
       print("Successfully verified document revision for id
'{}'!".format(metadata_id))
   else:
       print("Document revision for id '{}' verification
failed!".format(metadata_id))

   # Submit a request for the block and get a result back
   block_response = qldb_client.get_block(Name=ledger_name,
BlockAddress=block_address_to_dictionary(block_address),
                                          DigestTipAddress=digest_tip_address)

   block_text = block_response.get('Block').get('IonText')
   block = loads(block_text)

   block_hash = block.get('blockHash')

   proof_text = block_response.get('Proof').get('IonText')
   proof_hashes = loads(proof_text)

   # Calculate digest
   calculated_digest = reduce(dot, proof_hashes, block_hash)

   verified = calculated_digest == expected_digest
   if verified:
       print("Block address '{}' successfully
verified!".format(dumps(block_address,
                                                                  binary=False,

omit_version_marker=True)))
   else:
       print("Block address '{}' verification failed!".format(block_address))
```

Erreurs courantes de vérification

Cette section décrit les erreurs d'exécution émises par Amazon QLDB pour les demandes de vérification.

Voici une liste des exceptions courantes renvoyées par le service. Chaque exception inclut le message d'erreur spécifique, suivi des opérations d'API susceptibles de le générer, une brève description et des suggestions de solutions possibles.

IllegalArgumentException

Message : La valeur Ion fournie n'est pas valide et ne peut pas être analysée.

Opérations d'API : `GetDigest, GetBlock, GetRevision`

Assurez-vous de fournir une valeur Amazon Ion valide avant de réessayer votre demande.

IllegalArgumentException

Message : L'adresse de blocage fournie n'est pas valide.

Opérations d'API : `GetDigest, GetBlock, GetRevision`

Assurez-vous de fournir une adresse de blocage valide avant de réessayer votre demande. Une adresse de bloc est une structure Amazon Ion qui comporte deux champs : `strandId` et`sequenceNo`.

Par exemple : `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:14}`

IllegalArgumentException

Message : Le numéro de séquence de l'adresse de résumé fournie est supérieur au dernier enregistrement validé du fil.

Opérations d'API : `GetDigest, GetBlock, GetRevision`

L'adresse du résumé que vous fournissez doit avoir un numéro de séquence inférieur ou égal au numéro de séquence du dernier enregistrement validé du volet du journal. Avant de réessayer votre demande, assurez-vous de fournir une adresse de résumé avec un numéro de séquence valide.

IllegalArgumentException

Message : L'ID de brin de l'adresse de bloc fournie n'est pas valide.

Opérations d'API : `GetDigest, GetBlock, GetRevision`

L'adresse de bloc que vous fournissez doit avoir un identifiant de volet correspondant à l'identifiant de volet du journal. Avant de réessayer votre demande, assurez-vous de fournir une adresse de blocage avec un identifiant de chaîne valide.

IllegalArgumentException

Message : Le numéro de séquence de l'adresse de bloc fournie est supérieur au dernier enregistrement validé du brin.

Opérations d'API : `GetBlock, GetRevision`

L'adresse de bloc que vous fournissez doit avoir un numéro de séquence inférieur ou égal au numéro de séquence du dernier enregistrement validé du brin. Avant de réessayer votre demande, assurez-vous de fournir une adresse de blocage avec un numéro de séquence valide.

IllegalArgumentException

Message : L'ID de volet de l'adresse de bloc fournie doit correspondre à l'ID de volet de l'adresse de résumé fournie.

Opérations d'API : `GetBlock, GetRevision`

Vous ne pouvez vérifier une révision ou un bloc de document que s'il existe dans le même volet de journal que le résumé que vous fournissez.

IllegalArgumentException

Message : Le numéro de séquence de l'adresse de bloc fournie ne doit pas être supérieur au numéro de séquence de l'adresse de résumé fournie.

Opérations d'API : `GetBlock, GetRevision`

Vous ne pouvez vérifier une révision ou un bloc de document que s'il est couvert par le résumé que vous fournissez. Cela signifie qu'il a été enregistré dans le journal avant l'adresse du résumé.

IllegalArgumentException

Message : L'ID de document fourni n'a pas été trouvé dans le bloc à l'adresse de bloc spécifiée.

Fonctionnement de l'API : `GetRevision`

L'identifiant du document que vous fournissez doit figurer dans l'adresse de bloc que vous fournissez. Avant de réessayer votre demande, assurez-vous que ces deux paramètres sont cohérents.

Exportation de données de journal depuis Amazon QLDB

Amazon QLDB utilise un journal transactionnel immuable, appelé journal, pour le stockage des données. Le journal suit chaque modification apportée à vos données enregistrées et conserve un historique complet et vérifiable des modifications au fil du temps.

Vous pouvez accéder au contenu du journal dans votre registre à diverses fins, notamment pour l'analyse, l'audit, la conservation des données, la vérification et l'exportation vers d'autres systèmes. Les rubriques suivantes décrivent comment exporter des blocs de journaux de votre registre vers un compartiment Amazon Simple Storage Service (Amazon S3) situé dans votre. Compte AWS Une tâche d'exportation de journal écrit vos données dans Amazon S3 sous forme d'objets au format texte ou binaire d'Amazon Ion ou au format texte JSON Lines.

Au format JSON Lines, chaque bloc d'un objet de données exporté est un objet JSON valide délimité par une nouvelle ligne. Vous pouvez utiliser ce format pour intégrer directement les exportations JSON à des outils d'analyse tels qu'Amazon Athena et AWS Glue parce que ces services peuvent analyser automatiquement le JSON délimité par de nouvelles lignes. Pour plus d'informations sur le format, consultez Lignes JSON.

Pour plus d'informations sur Amazon S3, consultez le guide de l'utilisateur d'Amazon Simple Storage Service.

> **Note**
>
> Si vous spécifiez JSON comme format de sortie de votre tâche d'exportation, QLDB abaisse les données du journal Ion en JSON dans vos objets de données exportés. Pour plus d'informations, consultez Conversion descendante au format JSON.

Rubriques

Demande d'exportation d'un journal dans QLDB

Amazon QLDB fournit une API pour demander l'exportation de vos blocs de journal pour une date et une plage d'heures spécifiées et une destination de compartiment Amazon S3 spécifiée. Une tâche d'exportation de journal peut écrire les objets de données sous forme de texte ou de représentation binaire au format Amazon Ion, ou au format de texte JSON Lines. Vous pouvez utiliser le AWS Management Console, un AWS SDK ou le AWS Command Line Interface (AWS CLI) pour créer une tâche d'exportation.

Rubriques

AWS Management Console

Procédez comme suit pour soumettre une demande d'exportation de journal dans QLDB à l'aide de la console QLDB.

Pour demander une exportation (console)

1. Connectez-vous à la AWS Management Console console Amazon QLDB et ouvrez-la à l'adresse https://console.aws.amazon.com/qldb.
2. Dans le volet de navigation, choisissez Exporter.
3. Choisissez Créer une tâche d'exportation.
4. Sur la page Créer une tâche d'exportation, entrez les paramètres d'exportation suivants :
   - Ledger — Le registre dont vous souhaitez exporter les blocs de journal.
   - Date et heure de début : horodatage de début inclus en temps universel coordonné (UTC) de la plage de blocs de journal à exporter. Cet horodatage doit être antérieur à la date et à l'heure de fin. Si vous fournissez un horodatage de début antérieur à celui du registre, `CreationDateTime` QLDB le définit par défaut sur celui du registre. `CreationDateTime`
   - Date et heure de fin : horodatage de fin (UTC) exclusif de la plage de blocs de journal à exporter. Cette date et cette heure ne peuvent pas se situer dans le futur.

- Destination des blocs de journaux : nom du compartiment et du préfixe Amazon S3 dans lesquels votre tâche d'exportation écrit les objets de données. Utilisez le format d'URI Amazon S3 suivant.

  ```
  s3://DOC-EXAMPLE-BUCKET/prefix/
  ```

  Vous devez spécifier un nom de compartiment S3 et un nom de préfixe facultatif pour les objets de sortie. Voici un exemple.

  ```
  s3://DOC-EXAMPLE-BUCKET/journalExport/
  ```

  Le nom et le préfixe du compartiment doivent tous deux être conformes aux règles et conventions de dénomination d'Amazon S3. Pour plus d'informations sur la dénomination des compartiments, consultez la section Restrictions et limitations des compartiments dans le guide du développeur Amazon S3. Pour plus d'informations sur les préfixes des noms de clés, consultez la section Clé d'objet et métadonnées.

  > **Note**
  >
  > Les exportations entre régions ne sont pas prises en charge. Le compartiment Amazon S3 spécifié doit se trouver dans le même registre Région AWS que celui de votre registre.

- Chiffrement S3 : paramètres de chiffrement utilisés par votre tâche d'exportation pour écrire des données dans un compartiment Amazon S3. Pour en savoir plus sur les options de chiffrement côté serveur dans Amazon S3, consultez la section Protection des données à l'aide du chiffrement côté serveur dans le manuel Amazon S3 Developer Guide.
  - Chiffrement par défaut du compartiment : utilisez les paramètres de chiffrement par défaut du compartiment Amazon S3 spécifié.
  - AES-256 — Utilisez le chiffrement côté serveur avec les clés gérées par Amazon S3 (SSE-S3).
  - AWS-KMS — Utilisez le chiffrement côté serveur avec des clés AWS KMS gérées (SSE-KMS).

    Si vous choisissez ce type en même temps que l' AWS KMS key option Choisir un autre, vous devez également spécifier une clé KMS de chiffrement symétrique au format Amazon Resource Name (ARN) suivant.

```
arn:aws:kms:aws-region:account-id:key/key-id
```

- Accès au service : rôle IAM qui accorde les autorisations d'écriture QLDB dans votre compartiment Amazon S3. Le cas échéant, le rôle IAM doit également accorder des autorisations QLDB pour utiliser votre clé KMS.

  Pour transmettre un rôle à QLDB lors d'une demande d'exportation de journal, vous devez disposer des autorisations nécessaires pour effectuer *iam:PassRole* l'action sur la ressource du rôle IAM.

  - Création et utilisation d'un nouveau rôle de service : laissez la console créer un nouveau rôle pour vous avec les autorisations requises pour le compartiment Amazon S3 spécifié.
  - Utiliser un rôle de service existant : pour savoir comment créer manuellement ce rôle dans IAM, voirAutorisations d'exportation.
- Format de sortie : format de sortie des données de journal exportées
  - Texte ionique — (par défaut) Représentation textuelle d'Amazon Ion
  - Binaire ionique — Représentation binaire d'Amazon Ion
  - JSON — Format de texte JSON délimité par une nouvelle ligne

    Si vous choisissez JSON, QLDB abaisse les données du journal Ion en JSON dans vos objets de données exportés. Pour plus d'informations, consultez Conversion descendante au format JSON.

5. Lorsque les paramètres sont tels que vous le souhaitez, choisissez Créer une tâche d'exportation.

   Le temps nécessaire à la fin de votre tâche d'exportation varie en fonction de la taille des données. Si votre demande est soumise avec succès, la console revient à la page d'exportation principale et répertorie vos tâches d'exportation avec leur statut actuel.
6. Vous pouvez voir vos objets d'exportation sur la console Amazon S3.

   Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.

   Pour en savoir plus sur le format de ces objets de sortie, consultezSortie d'exportation du journal dans QLDB.

> **Note**
>
> Les tâches d'exportation expirent sept jours après leur fin. Pour plus d'informations, consultez Expiration du travail d'exportation.

API QLDB

Vous pouvez également demander une exportation de journal en utilisant l'API Amazon QLDB avec AWS un SDK ou le. AWS CLI L'API QLDB fournit les opérations suivantes destinées à être utilisées par les programmes d'application :

- `ExportJournalToS3`— Exporte le contenu du journal dans une plage de dates et d'heures à partir d'un registre donné vers un compartiment Amazon S3 spécifié. Une tâche d'exportation peut écrire les données sous forme d'objets au format texte ou binaire d'Amazon Ion, ou au format texte JSON Lines.
- `DescribeJournalS3Export`— Renvoie des informations détaillées sur une tâche d'exportation de journal. La sortie inclut son statut actuel, son heure de création et les paramètres de votre demande d'exportation initiale.
- `ListJournalS3Exports`— Renvoie une liste des descriptions de tâches d'exportation de journaux pour tous les registres associés au livre actuel Compte AWS et à la région. La sortie de chaque description de tâche d'exportation inclut les mêmes informations que celles renvoyées par`DescribeJournalS3Export`.
- `ListJournalS3ExportsForLedger`— Renvoie une liste des descriptions de tâches d'exportation de journaux pour un registre donné. La sortie de chaque description de tâche d'exportation inclut les mêmes informations que celles renvoyées par`DescribeJournalS3Export`.

Pour une description complète de ces opérations d'API, consultez leRéférence d'API Amazon QLDB.

Pour plus d'informations sur l'exportation de données de journal à l'aide du AWS CLI, consultez la référence des AWS CLI commandes.

Exemple d'application (Java)

Pour des exemples de code Java d'opérations d'exportation de base, consultez le GitHub référentiel amazon-qldb-dmv-sampleaws-samples/ -java. Pour obtenir des instructions sur le téléchargement

et l'installation de cet exemple d'application, consultezInstallation de l'exemple d'application Java Amazon QLDB. Avant de demander une exportation, assurez-vous de suivre les étapes 1 à 3 Tutoriel Java pour créer un registre d'échantillons et le charger avec des exemples de données.

Le code du didacticiel des classes suivantes fournit des exemples de création d'une exportation, de vérification de l'état d'une exportation et de traitement du résultat d'une exportation.

Classe	Description
ExportJournal	Exporte des blocs de journal depuis le registre `vehicle-registration` d'échantillons pour une plage d'horodatage allant d'il y a 10 minutes à aujourd'hui. Écrit les objets de sortie dans un compartiment S3 spécifié ou crée un compartiment unique s'il n'en est pas fourni.
DescribeJournalExport	Décrit une tâche d'exportation de journal pour un article spécifié `exportId` dans le registre `vehicle-registration` d'exemple.
ListJournalExports	Renvoie une liste des descriptions de tâches d'exportation de journaux pour le registre `vehicle-registration` d'exemple.
ValidateQldbHashChain	Valide la chaîne de hachage du registre d'`vehicle-registration` échantillons en utilisant une donnée. `exportId` Si ce n'est pas le cas, demande une nouvelle exportati on à utiliser pour la validation de la chaîne de hachage.

Expiration du travail d'exportation

Les tâches d'exportation de journaux terminées sont soumises à une période de conservation de 7 jours. Ils sont automatiquement supprimés définitivement après l'expiration de cette limite. Cette période d'expiration est une limite stricte et ne peut pas être modifiée.

Une fois qu'une tâche d'exportation terminée est supprimée, vous ne pouvez plus utiliser la console QLDB ou les opérations d'API suivantes pour récupérer les métadonnées relatives à la tâche :

- `DescribeJournalS3Export`
- `ListJournalS3Exports`
- `ListJournalS3ExportsForLedger`

Toutefois, cette expiration n'a aucun impact sur les données exportées elles-mêmes. Toutes les métadonnées sont conservées dans les fichiers manifestes écrits par vos exportations. Cette expiration est conçue pour faciliter les opérations d'API répertoriant les tâches d'exportation de journaux. QLDB supprime les anciennes tâches d'exportation afin que vous ne puissiez voir que les exportations récentes sans avoir à analyser plusieurs pages de tâches.

Sortie d'exportation du journal dans QLDB

Une tâche d'exportation de journal Amazon QLDB écrit deux fichiers manifestes en plus des objets de données qui contiennent vos blocs de journal. Ils sont tous enregistrés dans le compartiment Amazon S3 que vous avez fourni dans votre demande d'exportation. Les sections suivantes décrivent le format et le contenu de chaque objet en sortie.

> **Note**
>
> Si vous spécifiez JSON comme format de sortie de votre tâche d'exportation, QLDB convertit les données du journal Amazon Ion en JSON dans vos objets de données exportés. Pour plus d'informations, rendez-vous surConversion descendante au format JSON.

Rubriques

- Fichiers manifestes
- Objets de données
- Conversion descendante au format JSON
- bibliothèque de processeurs d'exportation (Java)

Fichiers manifestes

Amazon QLDB crée deux fichiers manifestes dans le compartiment S3 fourni pour chaque demande d'exportation. Le fichier manifeste initial est créé dès que vous soumettez la demande d'exportation. Le fichier manifeste final est écrit une fois l'exportation terminée. Vous pouvez utiliser ces fichiers pour vérifier le statut de vos tâches d'exportation dans Amazon S3.

Le format du contenu des fichiers manifestes correspond au format de sortie demandé pour l'exportation.

Manifeste initial

Le manifeste initial indique que votre tâche d'exportation a commencé. Il contient les paramètres d'entrée que vous avez transmis à la demande. Outre la destination Amazon S3 et les paramètres d'heure de début et de fin de l'exportation, ce fichier contient également un`exportId`. `exportId`Il s'agit d'un identifiant unique que QLDB attribue à chaque tâche d'exportation.

La convention de dénomination des fichiers est la suivante.

```
s3://DOC-EXAMPLE-BUCKET/prefix/exportId.started.manifest
```

Voici un exemple de fichier manifeste initial et de son contenu au format texte Ion.

```
s3://DOC-EXAMPLE-BUCKET/journalExport/8UyXulxccYLAsbN1aon7e4.started.manifest
```

```
{
  ledgerName:"my-example-ledger",
  exportId:"8UyXulxccYLAsbN1aon7e4",
  inclusiveStartTime:2019-04-15T00:00:00.000Z,
  exclusiveEndTime:2019-04-15T22:00:00.000Z,
  bucket:"DOC-EXAMPLE-BUCKET",
  prefix:"journalExport",
  objectEncryptionType:"NO_ENCRYPTION",
  outputFormat:"ION_TEXT"
}
```

Le manifeste initial inclut le manifeste `outputFormat` uniquement s'il a été spécifié dans la demande d'exportation. Si vous ne spécifiez pas le format de sortie, les données exportées sont `ION_TEXT` formatées par défaut.

Le fonctionnement de l'API DescribeJournalS3Export et le type de contenu des objets Amazon S3 exportés indiquent également le format de sortie.

Manifeste final

Le manifeste final indique que votre tâche d'exportation pour un volet de journal en particulier est terminée. La tâche d'exportation écrit un fichier manifeste final distinct pour chaque volet.

> **Note**
>
> Dans Amazon QLDB, un fil est une partition du journal de votre registre. QLDB prend actuellement en charge les revues à un seul volet.

Le manifeste final inclut une liste ordonnée de clés d'objets de données écrites lors de l'exportation. La convention de dénomination des fichiers est la suivante.

```
s3://DOC-EXAMPLE-BUCKET/prefix/exportId.strandId.completed.manifest
```

`strandId`Il s'agit d'un identifiant unique que QLDB attribue au brin. Voici un exemple de fichier manifeste final et de son contenu au format texte Ion.

```
s3://DOC-EXAMPLE-BUCKET/
journalExport/8UyXulxccYLAsbN1aon7e4.JdxjkR9bSYB5jMHWcI464T.completed.manifest
```

```
{
  keys:[
    "2019/04/15/22/JdxjkR9bSYB5jMHWcI464T.1-4.ion",
    "2019/04/15/22/JdxjkR9bSYB5jMHWcI464T.5-10.ion",
    "2019/04/15/22/JdxjkR9bSYB5jMHWcI464T.11-12.ion",
    "2019/04/15/22/JdxjkR9bSYB5jMHWcI464T.13-20.ion",
    "2019/04/15/22/JdxjkR9bSYB5jMHWcI464T.21-21.ion"
  ]
}
```

Objets de données

Amazon QLDB écrit les objets de données de journal dans le compartiment Amazon S3 fourni sous forme de texte ou de représentation binaire au format Amazon Ion, ou au format de texte JSON Lines.

Au format JSON Lines, chaque bloc d'un objet de données exporté est un objet JSON valide délimité par une nouvelle ligne. Vous pouvez utiliser ce format pour intégrer directement les exportations JSON à des outils d'analyse tels qu'Amazon Athena et AWS Glue parce que ces services peuvent analyser automatiquement le JSON délimité par de nouvelles lignes. Pour plus d'informations sur le format, consultez Lignes JSON.

Noms des objets de données

Une tâche d'exportation de journal écrit ces objets de données selon la convention de dénomination suivante.

```
s3://DOC-EXAMPLE-BUCKET/prefix/yyyy/mm/dd/hh/strandId.startSn-endSn.ion|.json
```

- Les données de sortie de chaque tâche d'exportation sont divisées en morceaux.
- `yyyy/mm/dd/hh`— La date et l'heure auxquelles vous avez soumis la demande d'exportation. Les objets exportés au cours de la même heure sont regroupés sous le même préfixe Amazon S3.
- `strandId`— L'identifiant unique du fil spécifique qui contient le bloc de journal en cours d'exportation.
- `startSn-endSn`— La plage de numéros de séquence incluse dans l'objet. Un numéro de séquence indique l'emplacement d'un bloc dans un fil.

Supposons, par exemple, que vous spécifiez le chemin suivant.

```
s3://DOC-EXAMPLE-BUCKET/journalExport/
```

Votre tâche d'exportation crée un objet de données Amazon S3 qui ressemble à ce qui suit. Cet exemple montre le nom d'un objet au format Ion.

```
s3://DOC-EXAMPLE-BUCKET/journalExport/2019/04/15/22/JdxjkR9bSYB5jMHWcI464T.1-5.ion
```

Contenu de l'objet de données

Chaque objet de données contient des objets de bloc de journal au format suivant.

```
{
  blockAddress: {
    strandId: String,
    sequenceNo: Int
```

```
  },
  transactionId: String,
  blockTimestamp: Datetime,
  blockHash: SHA256,
  entriesHash: SHA256,
  previousBlockHash: SHA256,
  entriesHashList: [ SHA256 ],
  transactionInfo: {
    statements: [
      {
        //PartiQL statement object
      }
    ],
    documents: {
      //document-table-statement mapping object
    }
  },
  revisions: [
    {
      //document revision object
    }
  ]
}
```

Un bloc est un objet qui est enregistré dans le journal lors d'une transaction. Un bloc contient des métadonnées de transaction ainsi que des entrées représentant les révisions du document qui ont été validées dans la transaction et les instructions partiQL qui les ont validées.

Voici un exemple de bloc contenant des exemples de données au format texte Ion. Pour plus d'informations sur les champs d'un objet de bloc, consultezContenu du journal dans Amazon QLDB.

> **Note**
>
> Cet exemple de bloc est fourni à titre informatif uniquement. Les hachages affichés ne sont pas de véritables valeurs de hachage calculées.

```
{
  blockAddress:{
    strandId:"JdxjkR9bSYB5jMHWcI464T",
    sequenceNo:1234
  },
```

```
  transactionId:"D35qctdJRU1L1N2VhxbwSn",
  blockTimestamp:2019-10-25T17:20:21.009Z,
  blockHash:{{WYLOfZClk0lYWT3lUsSr0ONXh+Pw8MxxB+9zvTgSvlQ=}},
  entriesHash:{{xN9X96atkMvhvF3nEy6jMSVQzKjHJfz1H3bsNeg8GMA=}},
  previousBlockHash:{{IAfZ0h22ZjvcuHPSBCDy/6XNQTsqEmeY3GW0gBae8mg=}},
  entriesHashList:[
      {{F7rQIKCNn0vXVWPexilGfJn5+MCrtsSQqqVdlQxXpS4=}},
      {{C+L8gRhkzVcxt3qRJpw8w6hVEqA5A6ImGne+E7iHizo=}}
  ],
  transactionInfo:{
    statements:[
      {
        statement:"CREATE TABLE VehicleRegistration",
        startTime:2019-10-25T17:20:20.496Z,
        statementDigest:{{3jeSdejOgp6spJ8huZxDRUtp2fRXRqpOMtG43V0nXg8=}}
      },
      {
        statement:"CREATE INDEX ON VehicleRegistration (VIN)",
        startTime:2019-10-25T17:20:20.549Z,
        statementDigest:{{099D+5ZWDgA7r+aWeNUrWhc8ebBTXjgscq+mZ2dVibI=}}
      },
      {
        statement:"CREATE INDEX ON VehicleRegistration (LicensePlateNumber)",
        startTime:2019-10-25T17:20:20.560Z,
        statementDigest:{{B73tVJzVyVXicnH4n96NzU2L2JFY8e9Tjg895suWMew=}}
      },
      {
        statement:"INSERT INTO VehicleRegistration ?",
        startTime:2019-10-25T17:20:20.595Z,
        statementDigest:{{ggpon5qCXLo95K578YVhAD8ix0A0M5CcBx/W40Ey/Tk=}}
      }
    ],
    documents:{
      '8F0TPCmdNQ6JTRpiLj2TmW':{
        tableName:"VehicleRegistration",
        tableId:"BPxNiDQXCIB5l5F68KZoOz",
        statements:[3]
      }
    }
  },
  revisions:[
    {
      hash:{{FR1IWcWew0yw1TnRklo2YMF/qtwb7ohsu5FD8A4DSVg=}}
    },
```

```
    {
      blockAddress:{
        strandId:"JdxjkR9bSYB5jMHWcI464T",
        sequenceNo:1234
      },
      hash:{{t8Hj6/VC4SBitxnvBqJbOmrGytF2XAA/1c0AoSq2NQY=}},
      data:{
        VIN:"1N4AL11D75C109151",
        LicensePlateNumber:"LEWISR261LL",
        State:"WA",
        City:"Seattle",
        PendingPenaltyTicketAmount:90.25,
        ValidFromDate:2017-08-21,
        ValidToDate:2020-05-11,
        Owners:{
          PrimaryOwner:{
            PersonId:"GddsXfIYfDlKCEprOLOwYt"
          },
          SecondaryOwners:[]
        }
      },
      metadata:{
        id:"8F0TPCmdNQ6JTRpiLj2TmW",
        version:0,
        txTime:2019-10-25T17:20:20.618Z,
        txId:"D35qctdJRU1L1N2VhxbwSn"
      }
    }
  ]
}
```

Sur le `revisions` terrain, certains objets de révision peuvent ne contenir qu'une `hash` valeur et aucun autre attribut. Il s'agit de révisions internes du système qui ne contiennent aucune donnée utilisateur. Une tâche d'exportation inclut ces révisions dans leurs blocs respectifs, car les hachages de ces révisions font partie de la chaîne de hachage complète du journal. La chaîne de hachage complète est requise pour la vérification cryptographique.

Conversion descendante au format JSON

Si vous spécifiez JSON comme format de sortie de votre tâche d'exportation, QLDB convertit les données du journal Amazon Ion en JSON dans vos objets de données exportés. Cependant, la

conversion d'Ion en JSON entraîne des pertes dans certains cas où vos données utilisent des types d'ions riches qui n'existent pas dans JSON.

Pour en savoir plus sur les règles de conversion d'ions en JSON, consultez la section Conversion descendante au format JSON dans le livre de recettes Amazon Ion.

bibliothèque de processeurs d'exportation (Java)

QLDB fournit un framework extensible pour Java qui rationalise le traitement des exportations dans Amazon S3. Cette bibliothèque de framework permet de lire le résultat d'une exportation et d'itérer les blocs exportés dans un ordre séquentiel. Pour utiliser ce processeur d'exportation, consultez le GitHub référentiel amazon-qldb-export-processorawslabs/ -java.

Autorisations d'exportation de journaux dans QLDB

Avant de soumettre une demande d'exportation de journal dans Amazon QLDB, vous devez fournir à QLDB des autorisations d'écriture dans le compartiment Amazon S3 que vous avez spécifié. Si vous choisissez un client géré AWS KMS key comme type de chiffrement d'objet pour votre compartiment Amazon S3, vous devez également autoriser QLDB à utiliser la clé de chiffrement symétrique que vous avez spécifiée. Amazon S3 ne prend pas en charge les clés KMS asymétriques.

Pour fournir à votre tâche d'exportation les autorisations nécessaires, vous pouvez faire en sorte que QLDB assume un rôle de service IAM avec les politiques d'autorisation appropriées. Une fonction de service est un rôle IAM qu'un service endosse pour accomplir des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer une fonction du service à partir d'IAM. Pour plus d'informations, consultez Création d'un rôle pour la délégation d'autorisations à un Service AWS dans le Guide de l'utilisateur IAM.

> **Note**
>
> Pour transmettre un rôle à QLDB lors d'une demande d'exportation de journal, vous devez disposer des autorisations nécessaires pour effectuer `iam:PassRole` l'action sur la ressource du rôle IAM. Cela s'ajoute à l'`qldb:ExportJournalToS3`autorisation sur la ressource du registre QLDB.
>
> Pour savoir comment contrôler l'accès à QLDB à l'aide d'IAM, consultez. Comment Amazon QLDB fonctionne avec IAM Pour un exemple de politique QLDB, consultez. Exemples de politiques basées sur l'identité pour Amazon QLDB

Dans cet exemple, vous créez un rôle qui permet à QLDB d'écrire des objets dans un compartiment Amazon S3 en votre nom. Pour plus d'informations, consultez Création d'un rôle pour la délégation d'autorisations à un Service AWS dans le Guide de l'utilisateur IAM.

Si vous exportez un journal QLDB dans Compte AWS votre journal pour la première fois, vous devez d'abord créer un rôle IAM avec les politiques appropriées en procédant comme suit. Vous pouvez également utiliser la console QLDB pour créer automatiquement le rôle pour vous. Sinon, vous pouvez choisir un rôle que vous avez créé précédemment.

Rubriques

- Créer une stratégie d'autorisations
- Créer un rôle IAM

Créer une stratégie d'autorisations

Procédez comme suit pour créer une politique d'autorisation pour une tâche d'exportation de journaux QLDB. Cet exemple montre une politique de compartiment Amazon S3 qui accorde à QLDB les autorisations d'écrire des objets dans le compartiment que vous avez spécifié. Le cas échéant, l'exemple montre également une politique de clé qui permet à QLDB d'utiliser votre clé KMS de chiffrement symétrique.

Pour plus d'informations sur les politiques relatives aux compartiments Amazon S3, consultez la section Utilisation des politiques relatives aux compartiments et des politiques utilisateur dans le guide de l'utilisateur d'Amazon Simple Storage Service. Pour en savoir plus sur les politiques AWS KMS clés, consultez la section Utilisation des politiques clés AWS KMS dans le Guide du AWS Key Management Service développeur.

> **Note**
>
> Votre compartiment Amazon S3 et votre clé KMS doivent tous deux se trouver dans le même registre Région AWS que votre registre QLDB.

Pour utiliser l'éditeur de politique JSON afin de créer une politique

1. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/iam/.

2. Dans la colonne de navigation de gauche, sélectionnez Politiques.

   Si vous choisissez Politiques pour la première fois, la page Bienvenue dans les politiques gérées s'affiche. Sélectionnez Mise en route.

3. En haut de la page, sélectionnez Créer une politique.

4. Sélectionnez l'onglet JSON.

5. Entrez un document de stratégie JSON.

   - Si vous utilisez une clé KMS gérée par le client pour le chiffrement des objets Amazon S3, utilisez l'exemple de document de politique suivant. *Pour utiliser cette politique, remplacez DOC-EXAMPLE-BUCKET, us-east-1, 123456789012 et 1234abcd-12ab-34cd-56ef-1234567890ab dans l'exemple par vos propres informations.*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBJournalExportS3Permission",
            "Action": [
                "s3:PutObjectAcl",
                "s3:PutObject"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        },
        {
            "Sid": "QLDBJournalExportKMSPermission",
            "Action": [ "kms:GenerateDataKey" ],
            "Effect": "Allow",
            "Resource": "arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ]
}
```

   - Pour les autres types de chiffrement, utilisez l'exemple de document de politique suivant. Pour utiliser cette politique, remplacez *DOC-EXAMPLE-BUCKET* dans l'exemple par votre propre nom de compartiment Amazon S3.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBJournalExportS3Permission",
            "Action": [
                "s3:PutObjectAcl",
                "s3:PutObject"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        }
    ]
}
```

6. Choisissez Examiner une politique.

> **Note**
>
> Vous pouvez basculer à tout moment entre les onglets Éditeur visuel et JSON. Toutefois, si vous apportez des modifications ou sélectionnez Examiner une politique dans l'onglet Éditeur visuel, IAM peut restructurer votre politique pour optimiser son affichage dans l'éditeur visuel. Pour de plus amples informations, consulte Restructuration d'une politique dans le Guide de l'utilisateur IAM.

7. Dans la page Examiner une politique, entrez un nom et éventuellement une description pour la politique que vous êtes en train de créer. Vérifiez le récapitulatif de la politique pour voir les autorisations accordées par votre politique. Sélectionnez ensuite Créer une politique pour enregistrer votre travail.

Créer un rôle IAM

Après avoir créé une politique d'autorisation pour votre tâche d'exportation de journal QLDB, vous pouvez créer un rôle IAM et y associer votre politique.

Pour créer le rôle de service pour QLDB (console IAM)

1. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation de la console IAM, sélectionnez Roles (Rôles), puis Create role (Créer un rôle).

3. Pour Trusted entity (Entité de confiance), choisissez Service AWS.
4. Pour Service ou cas d'utilisation, choisissez QLDB, puis choisissez le cas d'utilisation QLDB.
5. Choisissez Suivant.
6. Cochez la case à côté de la politique que vous avez créée lors des étapes précédentes.
7. (Facultatif) Définissez une limite d'autorisations. Il s'agit d'une fonctionnalité avancée disponible pour les fonctions de service, mais pas pour les rôles liés à un service.

   a. Ouvrez la section Définir les limites des autorisations, puis choisissez Utiliser une limite d'autorisations pour contrôler le nombre maximal d'autorisations de rôle.

      IAM inclut une liste des politiques AWS gérées et gérées par le client dans votre compte.

   b. Sélectionnez la politique à utiliser comme limite d'autorisations.
8. Choisissez Suivant.
9. Entrez un nom de rôle ou un suffixe de nom de rôle pour vous aider à identifier l'objectif du rôle.

   > **Important**
   >
   > Lorsque vous nommez un rôle, tenez compte des points suivants :
   >
   > - Les noms de rôles doivent être uniques au sein du Compte AWS vôtre et ne peuvent pas être rendus uniques au cas par cas.
   >
   >   Par exemple, ne créez pas de rôles nommés à la fois **`PRODROLE`** et**`prodrole`**. Lorsqu'un nom de rôle est utilisé dans une politique ou dans le cadre d'un ARN, il distingue les majuscules et minuscules, mais lorsqu'un nom de rôle apparaît aux clients dans la console, par exemple pendant le processus de connexion, le nom du rôle ne fait pas la distinction entre majuscules et minuscules.
   > - Vous ne pouvez pas modifier le nom du rôle une fois qu'il a été créé car d'autres entités peuvent y faire référence.
10. (Facultatif) Dans Description, entrez une description pour le rôle.
11. (Facultatif) Pour modifier les cas d'utilisation et les autorisations du rôle, dans les sections Étape 1 : Sélection des entités de confiance ou Étape 2 : Ajouter des autorisations, choisissez Modifier.
12. (Facultatif) Pour identifier, organiser ou rechercher le rôle, ajoutez des balises sous forme de paires clé-valeur. Pour plus d'informations sur l'utilisation des balises dans IAM, consultez la rubrique Balisage des ressources IAM dans le Guide de l'utilisateur IAM.

13. Passez en revue les informations du rôle, puis choisissez Create role (Créer un rôle).

Le document JSON suivant est un exemple de politique de confiance qui permet à QLDB d'assumer un rôle IAM auquel des autorisations spécifiques sont associées.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "qldb.amazonaws.com"
            },
            "Action": [ "sts:AssumeRole" ],
            "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:qldb:us-east-1:123456789012:*"
                },
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                }
            }
        }
    ]
}
```

> **Note**
>
> Cet exemple de politique de confiance montre comment vous pouvez utiliser les touches `aws:SourceArn` contextuelles et de condition `aws:SourceAccount` globale pour éviter le problème de confusion des adjoints. Grâce à cette politique de confiance, QLDB peut assumer le rôle de n'importe quelle ressource QLDB du compte uniquement. `123456789012` Pour plus d'informations, consultez Prévention du cas de figure de l'adjoint désorienté entre services.

Après avoir créé votre rôle IAM, revenez à la console QLDB et actualisez la page Créer une tâche d'exportation afin qu'elle puisse trouver votre nouveau rôle.

Erreurs courantes lors de l'exportation de journaux

Cette section décrit les erreurs d'exécution générées par Amazon QLDB pour les demandes d'exportation de journaux.

Voici une liste des exceptions courantes renvoyées par le service. Chaque exception inclut un message d'erreur spécifique, suivi d'une brève description et de suggestions de solutions possibles.

AccessDeniedException

> *Message : Utilisateur : UserArn n'est pas autorisé à exécuter : iam : PassRole on resource : Rolearn*
>
> Vous n'êtes pas autorisé à transmettre un rôle IAM au service QLDB. QLDB nécessite un rôle pour toutes les demandes d'exportation de journaux, et vous devez disposer des autorisations nécessaires pour transmettre ce rôle à QLDB. Le rôle fournit à QLDB des autorisations d'écriture dans le compartiment Amazon S3 que vous avez spécifié.
>
> Vérifiez que vous définissez une politique IAM qui autorise l'exécution de l'opération d'`PassRole`API sur la ressource de rôle IAM que vous avez spécifiée pour le service QLDB (). `qldb.amazonaws.com` Pour un exemple de stratégie, consultez Exemples de politiques basées sur l'identité pour Amazon QLDB.

IllegalArgumentException

> *Message : QLDB a rencontré une erreur lors de la validation de la configuration S3 : ErrorCode ErrorMessage*
>
> Cette erreur peut être due au fait que le compartiment Amazon S3 fourni n'existe pas dans Amazon S3. Ou bien, QLDB ne dispose pas des autorisations suffisantes pour écrire des objets dans le compartiment Amazon S3 que vous avez spécifié.
>
> Vérifiez que le nom du compartiment S3 que vous fournissez dans votre demande de tâche d'exportation est correct. Pour plus d'informations sur la dénomination des compartiments, consultez la section Restrictions et limitations des compartiments dans le guide de l'utilisateur d'Amazon Simple Storage Service.
>
> Vérifiez également que vous définissez une politique pour le compartiment que vous avez spécifié qui accorde `PutObject` des `PutObjectAcl` autorisations au service QLDB (). `qldb.amazonaws.com` Pour en savoir plus, veuillez consulter la section Autorisations d'exportation.

IllegalArgumentException

Message : réponse inattendue d'Amazon S3 lors de la validation de la configuration S3. *Réponse de S3 : ErrorCode ErrorMessage*

La tentative d'écriture des données d'exportation du journal dans le compartiment S3 fourni a échoué avec la réponse d'erreur Amazon S3 fournie. Pour plus d'informations sur les causes possibles, consultez la section Résolution des problèmes liés à Amazon S3 dans le guide de l'utilisateur d'Amazon Simple Storage Service.

IllegalArgumentException

Message : le préfixe du compartiment Amazon S3 ne doit pas dépasser 128 caractères

Le préfixe fourni dans la demande d'exportation du journal contient plus de 128 caractères.

IllegalArgumentException

Message : La date de début ne doit pas être supérieure à la date de fin

Les deux `InclusiveStartTime` `ExclusiveEndTime` doivent être au format de date et d'heure ISO 8601 et en temps universel coordonné (UTC).

IllegalArgumentException

Message : La date de fin ne peut pas être future

Les deux `InclusiveStartTime` `ExclusiveEndTime` doivent être au format `ISO 8601` date et heure et en UTC.

IllegalArgumentException

Message : Le paramètre de chiffrement d'objet fourni (S3EncryptionConfiguration) n'est pas compatible avec une clé AWS Key Management Service (AWS KMS)

Vous avez fourni `KMSKeyArn` l'un `ObjectEncryptionType` des deux `NO_ENCRYPTION` ou`SSE_S3`. Vous ne pouvez fournir qu'un client géré AWS KMS key pour un type de chiffrement d'objet de`SSE_KMS`. Pour en savoir plus sur les options de chiffrement côté serveur dans Amazon S3, consultez la section Protection des données à l'aide du chiffrement côté serveur dans le manuel Amazon S3 Developer Guide.

LimitExceededException

Message : Dépassement de la limite de 2 tâches d'exportation du Journal exécutées simultanément

QLDB impose une limite par défaut de deux tâches d'exportation de journaux simultanées.

Diffusion en continu de données de journaux depuis Amazon QLDB

Amazon QLDB utilise un journal transactionnel immuable, appelé journal, pour le stockage des données. Le journal suit chaque modification apportée à vos données enregistrées et conserve un historique complet et vérifiable des modifications au fil du temps.

Vous pouvez créer un flux dans QLDB qui capture chaque révision de document enregistrée dans votre journal et transmet ces données à Amazon Kinesis Data Streams en temps quasi réel. Le flux QLDB représente un flux ininterrompu de données allant du journal de votre registre vers une ressource de flux de données Kinesis.

Vous utilisez ensuite la plateforme de streaming Kinesis ou la bibliothèque cliente Kinesis pour utiliser votre flux, traiter les enregistrements de données et analyser le contenu des données. Un flux QLDB écrit vos données dans Kinesis Data Streams sous trois types d'enregistrements : contrôle, résumé des blocs et détails de révision. Pour plus d'informations, consultez Enregistrements de flux QLDB dans Kinesis.

Rubriques

Cas d’utilisation courants

Le streaming vous permet d'utiliser QLDB comme source fiable unique et vérifiable tout en intégrant les données de votre journal à d'autres services. Voici quelques-uns des cas d'utilisation courants pris en charge par les flux de journaux QLDB :

- Architecture axée sur les événements : créez des applications dans un style architectural axé sur les événements avec des composants découplés. Par exemple, une banque peut utiliser des AWS Lambda fonctions pour mettre en œuvre un système de notification qui alerte les clients lorsque le solde de leur compte tombe en dessous d'un seuil. Dans un tel système, les soldes des comptes sont conservés dans un registre QLDB, et tout changement de solde est enregistré dans le journal. La AWS Lambda fonction peut déclencher la logique de notification lors de la consommation d'un événement de mise à jour du solde qui est enregistré dans le journal et envoyé à un flux de données Kinesis.
- Analyses en temps réel : créez des applications grand public Kinesis qui exécutent des analyses en temps réel sur les données des événements. Grâce à cette fonctionnalité, vous pouvez obtenir des informations en temps quasi réel et réagir rapidement à l'évolution de l'environnement commercial. Par exemple, un site Web de commerce électronique peut analyser les données de vente de produits et arrêter les publicités pour un produit à prix réduit dès que les ventes atteignent une limite.
- Analyses historiques — Tirez parti de l'architecture orientée journal d'Amazon QLDB en relisant les données historiques des événements. Vous pouvez choisir de démarrer un flux QLDB à n'importe quel moment dans le passé, dans lequel toutes les révisions effectuées depuis lors sont transmises à Kinesis Data Streams. Grâce à cette fonctionnalité, vous pouvez créer des applications grand public Kinesis qui exécutent des tâches d'analyse sur des données historiques. Par exemple, un site Web de commerce électronique peut exécuter des analyses selon les besoins pour générer des statistiques de ventes passées qui n'avaient pas été capturées auparavant.
- Réplication vers des bases de données spécialement conçues : connectez les registres QLDB à d'autres magasins de données spécialisés à l'aide des flux de journaux QLDB. Par exemple, utilisez la plateforme de données de streaming Kinesis pour intégrer Amazon OpenSearch Service, qui peut fournir des fonctionnalités de recherche en texte intégral pour les documents QLDB. Vous pouvez également créer des applications Kinesis personnalisées pour répliquer les données de votre journal dans d'autres bases de données spécialement conçues qui fournissent différentes vues matérialisées. Répliquez, par exemple, vers Amazon Aurora pour les données relationnelles ou vers Amazon Neptune pour les données basées sur des graphes.

Consommation de votre stream

Utilisez Kinesis Data Streams pour consommer, traiter et analyser en continu de grands flux d'enregistrements de données. Outre Kinesis Data Streams, la plateforme de données de streaming Kinesis inclut Amazon Data Firehose et Amazon Managed Service pour Apache Flink. Vous pouvez utiliser cette plate-forme pour envoyer des enregistrements de données directement à des services tels qu'Amazon OpenSearch Service, Amazon Redshift, Amazon S3 ou Splunk. Pour plus d'informations, consultez la section destinée aux utilisateurs de Kinesis Data Streams dans le manuel Amazon Kinesis Data Streams Developer Guide.

Vous pouvez également utiliser la bibliothèque client Kinesis (KCL) pour créer une application client de streaming afin de traiter les enregistrements de données de manière personnalisée. La KCL simplifie le codage en fournissant des abstractions utiles par-dessus l'API Kinesis Data Streams de bas niveau. Pour en savoir plus sur la KCL, consultez la section Utilisation de la bibliothèque cliente Kinesis dans le guide du développeur Amazon Kinesis Data Streams.

Garantie de livraison

Les flux QLDB offrent at-least-onceune garantie de livraison. Chaque enregistrement de données produit par un flux QLDB est transmis à Kinesis Data Streams au moins une fois. Les mêmes enregistrements peuvent apparaître plusieurs fois dans un flux de données Kinesis. Vous devez donc disposer d'une logique de déduplication dans la couche d'application grand public si votre cas d'utilisation l'exige.

Il n'y a pas non plus de garantie de commande. Dans certains cas, des blocs et des révisions QLDB peuvent être produits dans un flux de données Kinesis de manière désordonnée. Pour plus d’informations, consultez Gestion des doublons et out-of-order des enregistrements.

Considérations relatives au délai de livraison

Les flux QLDB fournissent généralement des mises à jour de Kinesis Data Streams en temps quasi réel. Toutefois, les scénarios suivants peuvent créer une latence supplémentaire avant que les données QLDB nouvellement validées ne soient émises vers un flux de données Kinesis :

- Kinesis peut limiter les données diffusées depuis QLDB, en fonction de votre approvisionnement en Kinesis Data Streams. Cela peut se produire, par exemple, si plusieurs flux QLDB écrivent dans un seul flux de données Kinesis et que le taux de demandes de QLDB dépasse la capacité de la ressource de flux Kinesis. La limitation dans Kinesis peut également se produire lors de

l'utilisation du provisionnement à la demande si le débit augmente pour atteindre plus du double du pic précédent en moins de 15 minutes.

Vous pouvez mesurer ce débit dépassé en surveillant la métrique `WriteProvisionedThroughputExceeded` Kinesis. Pour plus d'informations et des solutions possibles, consultez Comment résoudre les erreurs de régulation dans Kinesis Data Streams ? .

- Avec les flux QLDB, vous pouvez créer un flux indéfini dont la date et l'heure de début sont anciennes et sans date ni heure de fin. De par sa conception, QLDB ne commence à émettre des données nouvellement validées vers Kinesis Data Streams qu'une fois que toutes les données antérieures à la date et à l'heure de début spécifiées ont été transmises avec succès. Si vous constatez une latence supplémentaire dans ce scénario, vous devrez peut-être attendre que les données précédentes soient livrées, ou vous pouvez démarrer le flux à une date et une heure de début ultérieures.

Commencer à utiliser les streams

Vous trouverez ci-dessous une présentation détaillée des étapes nécessaires pour commencer à diffuser des données de journal vers Kinesis Data Streams :

1. Créez une ressource Kinesis Data Streams. Pour obtenir des instructions, consultez la section Création et mise à jour de flux de données dans le manuel Amazon Kinesis Data Streams Developer Guide.
2. Créez un rôle IAM qui permet à QLDB d'assumer des autorisations d'écriture pour le flux de données Kinesis. Pour obtenir des instructions, veuillez consulter Autorisations de diffusion dans QLDB.
3. Créez un flux de journal QLDB. Pour obtenir des instructions, veuillez consulter Création et gestion de flux dans QLDB.
4. Consommez le flux de données Kinesis, comme décrit dans la section précédente. Consommation de votre stream Pour des exemples de code illustrant l'utilisation de la bibliothèque cliente Kinesis AWS Lambda, voir. Développement avec des flux dans QLDB

Création et gestion de flux dans QLDB

Amazon QLDB fournit des opérations d'API pour créer et gérer un flux de données de journal depuis votre registre vers Amazon Kinesis Data Streams. Le flux QLDB capture chaque révision de document validée dans votre journal et l'envoie vers un flux de données Kinesis.

Vous pouvez utiliser le AWS Management Console, un AWS SDK ou le AWS Command Line Interface (AWS CLI) pour créer un flux de journal. En outre, vous pouvez également utiliser un AWS CloudFormationmodèle pour créer des flux. Pour plus d'informations, consultez la AWS::QLDB::Streamressource dans le guide de AWS CloudFormation l'utilisateur.

Rubriques

Paramètres du flux

Pour créer un flux de journal QLDB, vous devez fournir les paramètres de configuration suivants :

Nom du registre

Le registre QLDB dont vous souhaitez diffuser les données du journal vers Kinesis Data Streams.

Nom du flux

Nom que vous souhaitez affecter au flux de journal QLDB. Les noms définis par l'utilisateur peuvent aider à identifier et à indiquer le but d'un flux.

Votre nom de flux doit être unique parmi les autres flux actifs pour un registre donné. Les noms de flux sont soumis aux mêmes contraintes de dénomination que les noms de registre, tels que définis dansQuotas et limites d'Amazon QLDB.

Outre le nom du flux, QLDB attribue un ID de flux à chaque flux QLDB que vous créez. L'ID de flux est unique parmi tous les flux pour un registre donné, quel que soit leur statut.

Date et heure de début

Date et heure à partir desquelles commencer à diffuser les données du journal. Cette valeur peut être n'importe quelle date et heure dans le passé, mais pas dans le futur.

Date et heure de fin

(Facultatif) Date et heure qui indiquent la fin du flux.

Si vous créez un flux indéfini sans heure de fin, vous devez l'annuler manuellement pour terminer le flux. Vous pouvez également annuler un flux actif et limité qui n'a pas encore atteint la date et l'heure de fin spécifiées.

Flux de données Kinesis de destination

La ressource cible Kinesis Data Streams dans laquelle votre flux écrit les enregistrements de données. Pour savoir comment créer un flux de données Kinesis, consultez la section Création et mise à jour de flux de données dans le manuel Amazon Kinesis Data Streams Developer Guide.

> ⚠ Important
>
> - Les flux entre régions et entre comptes ne sont pas pris en charge. Le flux de données Kinesis spécifié doit se trouver dans le même Région AWS compte que votre registre.
> - L'agrégation d'enregistrements dans Kinesis Data Streams est activée par défaut. Cette option permet à QLDB de publier plusieurs enregistrements de données dans un seul enregistrement Kinesis Data Streams, augmentant ainsi le nombre d'enregistrements envoyés par appel d'API.
>
> L'agrégation d'enregistrements a des implications importantes pour le traitement des enregistrements et nécessite une désagrégation chez votre consommateur de flux. Pour en savoir plus, consultez les concepts clés du KPL et la désagrégation des consommateurs dans le manuel Amazon Kinesis Data Streams Developer Guide.

Rôle IAM

Rôle IAM qui permet à QLDB d'attribuer des autorisations d'écriture à votre flux de données Kinesis. Vous pouvez utiliser la console QLDB pour créer automatiquement ce rôle, ou vous pouvez le créer manuellement dans IAM. Pour savoir comment le créer manuellement, voirAutorisations de diffusion.

Pour transmettre un rôle à QLDB lorsque vous demandez un flux de journal, vous devez disposer des autorisations nécessaires pour effectuer l'action `iam:PassRole` sur la ressource de rôle IAM.

ARN du flux de diffusion

Chaque flux de journal QLDB est une sous-ressource d'un registre et est identifié de manière unique par un Amazon Resource Name (ARN). Voici un exemple d'ARN d'un flux QLDB dont l'ID de flux est pour un registre `IiPT4brpZCqCq3f4MTHbYy` nommé. `exampleLedger`

```
arn:aws:qldb:us-east-1:123456789012:stream/exampleLedger/IiPT4brpZCqCq3f4MTHbYy
```

La section suivante décrit comment créer et annuler un flux QLDB à l'aide du. AWS Management Console

AWS Management Console

Procédez comme suit pour créer ou annuler un flux QLDB à l'aide de la console QLDB.

Pour créer un flux (console)

1. Connectez-vous à la AWS Management Console console Amazon QLDB et ouvrez-la à l'adresse https://console.aws.amazon.com/qldb.
2. Choisissez Streams (Flux) dans le panneau de navigation.
3. Choisissez Create QLDB stream.
4. Sur la page Créer un flux QLDB, entrez les paramètres suivants :

   - Nom du flux : nom que vous souhaitez attribuer au flux QLDB.
   - Ledger — Le registre dont vous souhaitez diffuser les données du journal.
   - Date et heure de début : horodatage inclus en temps universel coordonné (UTC) à partir duquel commencer à diffuser les données du journal. Cet horodatage correspond par défaut à la date et à l'heure actuelles. Cela ne peut pas être dans le futur et doit être antérieur à la date et à l'heure de fin.
   - Date et heure de fin — (Facultatif) Horodatage exclusif (UTC) qui indique la fin du flux. Si vous laissez ce paramètre vide, le flux s'exécute indéfiniment jusqu'à ce que vous l'annuliez.
   - Flux de destination : ressource cible Kinesis Data Streams dans laquelle votre flux écrit les enregistrements de données. Utilisez le format ARN suivant.

     ```
     arn:aws:kinesis:aws-region:account-id:stream/kinesis-stream-name
     ```

     Voici un exemple.

```
arn:aws:kinesis:us-east-1:123456789012:stream/stream-for-qldb
```

Les flux entre régions et entre comptes ne sont pas pris en charge. Le flux de données Kinesis spécifié doit se trouver dans le même Région AWS compte que votre registre.

- Activer l'agrégation d'enregistrements dans Kinesis Data Streams : (Activé par défaut) Permet à QLDB de publier plusieurs enregistrements de données dans un seul enregistrement Kinesis Data Streams, augmentant ainsi le nombre d'enregistrements envoyés par appel d'API.
- Accès au service : rôle IAM qui accorde à QLDB des autorisations d'écriture sur votre flux de données Kinesis.

  Pour transmettre un rôle à QLDB lorsque vous demandez un flux de journal, vous devez disposer des autorisations nécessaires pour effectuer *`iam:PassRole`* l'action sur la ressource du rôle IAM.

  - Créer et utiliser un nouveau rôle de service : laissez la console créer un nouveau rôle pour vous avec les autorisations requises pour le flux de données Kinesis spécifié.
  - Utiliser un rôle de service existant : pour savoir comment créer manuellement ce rôle dans IAM, voirAutorisations de diffusion.
- Balises — (Facultatif) Ajoutez des métadonnées au flux en attachant des balises sous forme de paires clé-valeur. Vous pouvez ajouter des tags à votre stream pour mieux les organiser et les identifier. Pour plus d'informations, consultez Balisage des ressources Amazon QLDB.

  Choisissez Ajouter une balise, puis entrez les paires clé-valeur appropriées.

5. Lorsque les paramètres sont tels que vous le souhaitez, choisissez Create QLDB stream.

   Si l'envoi de votre demande aboutit, la console revient à la page principale des flux et répertorie vos flux QLDB avec leur statut actuel.

6. Une fois votre flux actif, utilisez Kinesis pour traiter les données de votre flux avec une application grand public.

   Ouvrez la console Kinesis Data Streams à l'adresse https://console.aws.amazon.com/kinesis/.

Pour plus d'informations sur le format des enregistrements de données de flux, consultezEnregistrements de flux QLDB dans Kinesis.

Pour savoir comment gérer les flux qui génèrent une erreur, consultezGestion des flux altérés.

Pour annuler un stream (console)

Vous ne pouvez pas redémarrer un flux QLDB après l'avoir annulé. Pour reprendre la livraison de vos données vers Kinesis Data Streams, vous pouvez créer un nouveau flux QLDB.

1. Ouvrez la console Amazon QLDB à l'adresse https://console.aws.amazon.com/qldb.
2. Choisissez Streams (Flux) dans le panneau de navigation.
3. Dans la liste des flux QLDB, sélectionnez le flux actif que vous souhaitez annuler.
4. Choisissez Annuler le stream. Confirmez cela en entrant **`cancel stream`** dans le champ prévu à cet effet.

Pour plus d'informations sur l'utilisation de l'API QLDB avec AWS un SDK ou sur la création et AWS CLI la gestion de flux de journaux, consultez. Développement avec des flux dans QLDB

États du flux

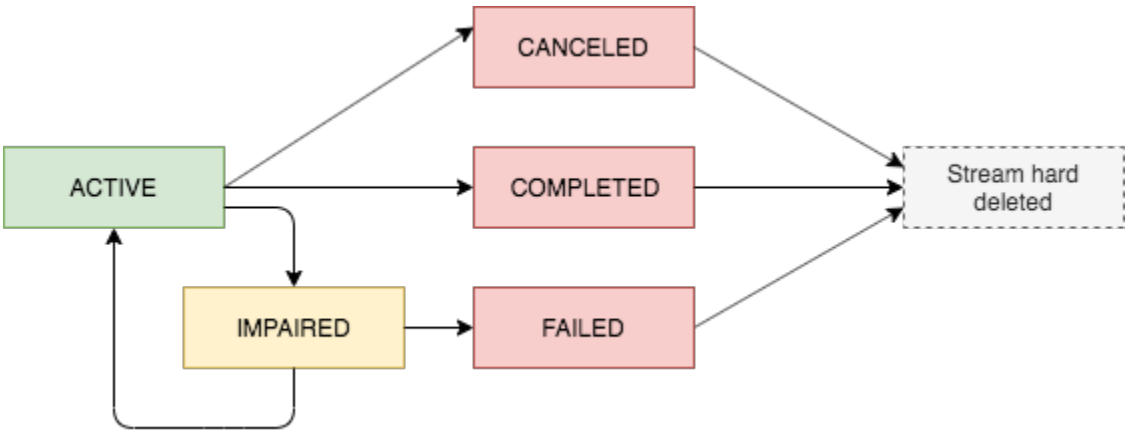
L'état d'un flux QLDB peut être l'un des suivants :

- `ACTIVE`— Diffuse actuellement ou attend de diffuser des données (pour un flux indéfini sans heure de fin).
- `COMPLETED`— A terminé avec succès la diffusion en continu de tous les blocs de journal dans la plage de temps spécifiée. Il s’agit d’un statut de terminal.

- `CANCELED`— A été interrompue par une demande utilisateur avant l'heure de fin spécifiée et ne diffuse plus activement de données. Il s’agit d’un statut de terminal.
- `IMPAIRED`— Impossible d'écrire des enregistrements dans Kinesis en raison d'une erreur nécessitant votre intervention. Il s'agit d'un état non terminal récupérable.

  Si vous corrigez l'erreur dans un délai d'une heure, le flux passe automatiquement à `ACTIVE` l'état. Si l'erreur n'est toujours pas résolue au bout d'une heure, le flux passe automatiquement à `FAILED` l'état.
- `FAILED`— Impossible d'écrire des enregistrements dans Kinesis en raison d'une erreur et se trouve dans un état terminal irrécupérable.

Le schéma suivant illustre comment une ressource de flux QLDB peut passer d'un état à l'autre.



Expiration des flux du terminal

Les ressources de streaming qui sont dans un état terminal (`CANCELED`COMPLETED, et`FAILED`) sont soumises à une période de conservation de 7 jours. Ils sont automatiquement supprimés définitivement après l'expiration de cette limite.

Après la suppression d'un flux de terminal, vous ne pouvez plus utiliser la console QLDB ou l'API QLDB pour décrire ou répertorier la ressource du flux.

Gestion des flux altérés

Si votre stream rencontre une erreur, il passe d'abord à `IMPAIRED` l'état. QLDB continue de `IMPAIRED` réessayer des streams pendant une heure maximum.

Si vous corrigez l'erreur dans un délai d'une heure, le flux passe automatiquement à `ACTIVE` l'état. Si l'erreur n'est toujours pas résolue au bout d'une heure, le flux passe automatiquement à `FAILED` l'état.

Un flux altéré ou défaillant peut avoir l'une des causes d'erreur suivantes :

- `KINESIS_STREAM_NOT_FOUND`— La ressource Kinesis Data Streams de destination n'existe pas. Vérifiez que le flux de données Kinesis que vous avez fourni dans votre demande de flux QLDB est correct. Accédez ensuite à Kinesis et créez le flux de données que vous avez spécifié.
- `IAM_PERMISSION_REVOKED`— QLDB ne dispose pas des autorisations suffisantes pour écrire des enregistrements de données dans le flux de données Kinesis que vous avez spécifié. Vérifiez que vous définissez une politique pour le flux de données Kinesis que vous avez spécifié qui accorde au service QLDB (`qldb.amazonaws.com`) des autorisations pour les actions suivantes :
  - `kinesis:PutRecord`

- `kinesis:PutRecords`
- `kinesis:DescribeStream`
- `kinesis:ListShards`

Surveillance des flux altérés

Si un flux est altéré, la console QLDB affiche une bannière contenant des informations détaillées sur le flux et l'erreur qu'il a rencontrée. Vous pouvez également utiliser l'opération `DescribeJournalKinesisStream` API pour obtenir le statut d'un flux et la cause de l'erreur sous-jacente.

En outre, vous pouvez utiliser Amazon CloudWatch pour créer une alarme qui surveille la `IsImpaired` métrique d'un flux. Pour plus d'informations sur la surveillance des métriques CloudWatch QLDB avec, consultez. Dimensions et statistiques d'Amazon QLDB

Développement avec des flux dans QLDB

Cette section résume les opérations d'API que vous pouvez utiliser avec un AWS SDK ou AWS CLI pour créer et gérer des flux de journaux dans Amazon QLDB. Il décrit également les exemples d'applications qui illustrent ces opérations et utilisent la Kinesis Client Library (KCL) ou AWS Lambda pour implémenter un consommateur de flux.

Vous pouvez utiliser la KCL pour créer des applications grand public pour Amazon Kinesis Data Streams. La KCL simplifie le codage en fournissant des abstractions utiles par-dessus l'API Kinesis Data Streams de bas niveau. Pour en savoir plus sur la KCL, consultez la section Utilisation de la bibliothèque cliente Kinesis dans le guide du développeur Amazon Kinesis Data Streams.

Table des matières

API de flux de journaux QLDB

L'API QLDB fournit les opérations de flux de journal suivantes destinées aux programmes d'application :

- `StreamJournalToKinesis`— Crée un flux de journal pour un registre QLDB donné. Le flux capture toutes les révisions de documents enregistrées dans le journal du registre et transmet les données à une ressource Kinesis Data Streams spécifiée.
  - L'agrégation d'enregistrements dans Kinesis Data Streams est activée par défaut. Cette option permet à QLDB de publier plusieurs enregistrements de données dans un seul enregistrement Kinesis Data Streams, augmentant ainsi le nombre d'enregistrements envoyés par appel d'API.

    L'agrégation d'enregistrements a des implications importantes pour le traitement des enregistrements et nécessite une désagrégation chez votre consommateur de flux. Pour en savoir plus, consultez les concepts clés du KPL et la désagrégation des consommateurs dans le manuel Amazon Kinesis Data Streams Developer Guide.
- `DescribeJournalKinesisStream`— Renvoie des informations détaillées sur un flux de journal QLDB donné. La sortie inclut l'ARN, le nom du flux, l'état actuel, l'heure de création et les paramètres de votre demande de création de flux d'origine.
- `ListJournalKinesisStreamsForLedger`— Renvoie une liste de tous les descripteurs de flux de journaux QLDB pour un registre donné. La sortie de chaque descripteur de flux inclut les mêmes détails que ceux renvoyés par`DescribeJournalKinesisStream`.
- `CancelJournalKinesisStream`— Termine un flux de journal QLDB donné. Avant qu'un stream puisse être annulé, son statut actuel doit être`ACTIVE`.

  Vous ne pouvez pas redémarrer un stream après l'avoir annulé. Pour reprendre la livraison de vos données vers Kinesis Data Streams, vous pouvez créer un nouveau flux QLDB.

Pour une description complète de ces opérations d'API, consultez leRéférence d'API Amazon QLDB.

Pour plus d'informations sur la création et la gestion de flux de journaux à l'aide du AWS CLI, consultez la référence des AWS CLI commandes.

Exemples d’applications

QLDB fournit des exemples d'applications illustrant diverses opérations à l'aide de flux de journaux. Ces applications sont open source sur le GitHub site AWS Samples.

Rubriques

- Opérations de base (Java)
- Intégration avec OpenSearch le service (Python)
- Intégration avec Amazon SNS et Amazon SQS (Python)

Opérations de base (Java)

Pour un exemple de code Java illustrant les opérations de base pour les flux de journaux QLDB, consultez GitHub le référentiel aws-samples/ -java. amazon-qldb-dmv-sample Pour obtenir des instructions sur le téléchargement et l'installation de cet exemple d'application, consultezInstallation de l'exemple d'application Java Amazon QLDB.

> **Note**
>
> Après avoir installé l'application, ne passez pas à l'étape 1 du didacticiel Java pour créer un registre. Cet exemple d'application pour le streaming crée le `vehicle-registration` registre pour vous.

Cet exemple d'application contient le code source complet à partir de Tutoriel Java et de ses dépendances, y compris les modules suivants :

- AWS SDK for Java— Pour créer et supprimer les ressources QLDB et Kinesis Data Streams, notamment les registres, les flux de journaux QLDB et les flux de données Kinesis.
- Pilote Amazon QLDB pour Java— Pour exécuter des transactions de données sur un registre à l'aide d'instructions partiQL, notamment en créant des tables et en insérant des documents.
- Bibliothèque cliente Kinesis : pour consommer et traiter les données d'un flux de données Kinesis.

Exécution du code

La StreamJournalclasse contient un code didacticiel illustrant les opérations suivantes :

1. Créez un registre nommévehicle-registration, créez des tables et chargez-les avec des exemples de données.

> **Note**
>
> Avant d'exécuter ce code, assurez-vous qu'aucun registre actif n'est déjà nommévehicle-registration.

2. Créez un flux de données Kinesis, un rôle IAM qui permet à QLDB d'assumer des autorisations d'écriture pour le flux de données Kinesis et un flux de journal QLDB.
3. Utilisez la KCL pour démarrer un lecteur de flux qui traite le flux de données Kinesis et enregistre chaque enregistrement de données QLDB.
4. Utilisez les données du flux pour valider la chaîne de hachage du registre d'`vehicle-registration`échantillons.
5. Nettoyez toutes les ressources en arrêtant le lecteur de flux, en annulant le flux de journal QLDB, en supprimant le registre et en supprimant le flux de données Kinesis.

Pour exécuter le code du `StreamJournal` didacticiel, entrez la commande Gradle suivante depuis le répertoire racine de votre projet.

```
./gradlew run -Dtutorial=streams.StreamJournal
```

Intégration avec OpenSearch le service (Python)

Pour un exemple d'application Python qui montre comment intégrer un flux QLDB à OpenSearch Amazon Service, consultez GitHub le référentiel aws-samples/ -. amazon-qldb-streaming-amazon opensearch-service-sample-python Cette application utilise une AWS Lambda fonction pour implémenter un consommateur Kinesis Data Streams.

Pour cloner le référentiel, entrez la `git` commande suivante.

```
git clone https://github.com/aws-samples/amazon-qldb-streaming-amazon-opensearch-
service-sample-python.git
```

Pour exécuter l'exemple d'application, consultez le fichier README activé GitHub pour obtenir des instructions.

Intégration avec Amazon SNS et Amazon SQS (Python)

Pour un exemple d'application Python qui montre comment intégrer un flux QLDB à Amazon Simple Notification Service (Amazon SNS), consultez le référentiel aws-samples/ -. GitHub amazon-qldb-streams-dmv sample-lambda-python

Cette application utilise une AWS Lambda fonction pour implémenter un consommateur Kinesis Data Streams. Il envoie des messages à une rubrique Amazon SNS à laquelle une file d'attente Amazon Simple Queue Service (Amazon SQS) est abonnée.

Pour cloner le référentiel, entrez la `git` commande suivante.

```
git clone https://github.com/aws-samples/amazon-qldb-streams-dmv-sample-lambda-
python.git
```

Pour exécuter l'exemple d'application, consultez le fichier README activé GitHub pour obtenir des instructions.

Enregistrements de flux QLDB dans Kinesis

Un flux Amazon QLDB écrit trois types d'enregistrements de données sur une ressource Amazon Kinesis Data Streams donnée : contrôle, résumé des blocs et détails de révision. Les trois types d'enregistrement sont écrits dans la représentation binaire du format Amazon Ion.

Les enregistrements de contrôle indiquent le début et la fin de vos flux QLDB. Chaque fois qu'une révision est validée dans votre journal, un flux QLDB écrit toutes les données de bloc de journal associées dans des enregistrements de résumé des blocs et de détails de révision.

Les trois types d'enregistrement sont polymorphes. Ils consistent tous en un enregistrement de haut niveau commun qui contient l'ARN du flux QLDB, le type d'enregistrement et la charge utile de l'enregistrement. Cet enregistrement de haut niveau a le format suivant.

```
{
  qldbStreamArn: string,
  recordType: string,
  payload: {
    //control | block summary | revision details record
  }
}
```

Le `recordType` champ peut avoir l'une des trois valeurs suivantes :

- `CONTROL`
- `BLOCK_SUMMARY`
- `REVISION_DETAILS`

Les sections suivantes décrivent le format et le contenu de chaque enregistrement de charge utile individuel.

> **Note**
>
> QLDB écrit tous les enregistrements de flux dans Kinesis Data Streams dans la représentation binaire d'Amazon Ion. Les exemples suivants sont fournis dans la représentation textuelle d'Ion pour illustrer le contenu de l'enregistrement dans un format lisible.

Rubriques

Enregistrements de contrôle

Un flux QLDB écrit des enregistrements de contrôle pour indiquer ses événements de début et de fin. Voici des exemples d'enregistrements de contrôle avec des échantillons de données pour chacun d'entre eux `controlRecordType` :

- `CREATED`— Le premier enregistrement qu'un flux QLDB écrit dans Kinesis pour indiquer que le flux que vous venez de créer est actif.

```
{
  qldbStreamArn:"arn:aws:qldb:us-east-1:123456789012:stream/exampleLedger/
IiPT4brpZCqCq3f4MTHbYy",
  recordType:"CONTROL",
  payload:{
```

```
    controlRecordType:"CREATED"
  }
}
```

- `COMPLETED`— Le dernier enregistrement qu'un flux QLDB écrit dans Kinesis pour indiquer que votre flux a atteint la date et l'heure de fin spécifiées. Cet enregistrement n'est pas écrit si vous annulez le stream.

```
{
  qldbStreamArn:"arn:aws:qldb:us-east-1:123456789012:stream/exampleLedger/
IiPT4brpZCqCq3f4MTHbYy",
  recordType:"CONTROL",
  payload:{
    controlRecordType:"COMPLETED"
  }
}
```

Bloquer les enregistrements récapitulatifs

Un enregistrement récapitulatif de bloc représente un bloc de journal dans lequel les révisions de votre document sont validées. Un bloc est un objet qui est enregistré dans votre journal QLDB lors d'une transaction.

La charge utile d'un enregistrement récapitulatif de bloc contient l'adresse du bloc, l'horodatage et les autres métadonnées de la transaction qui a validé le bloc. Il inclut également les attributs récapitulatifs des révisions du bloc et les instructions partiQL qui les ont validées. Voici un exemple d'enregistrement récapitulatif en bloc avec des exemples de données.

> **Note**
>
> Cet exemple de résumé de bloc est fourni à titre informatif uniquement. Les hachages affichés ne sont pas de véritables valeurs de hachage calculées.

```
{
  qldbStreamArn:"arn:aws:qldb:us-east-1:123456789012:stream/exampleLedger/
IiPT4brpZCqCq3f4MTHbYy",
  recordType:"BLOCK_SUMMARY",
  payload:{
    blockAddress:{
```

```
      strandId:"ElYL30RGoqrFCbbaQn3K6m",
      sequenceNo:60807
    },
    transactionId:"9RWohCo7My4GGkxRETAJ6M",
    blockTimestamp:2019-09-18T17:00:14.601000001Z,
    blockHash:{{6Pk9KDYJd38ci09oaHxx0D2grtgh4QBBqbDS6i9quX8=}},
    entriesHash:{{r5YoH6+NXDXxgoRzPREGAWJfn73KlZE0eTfbTxZWUDU=}},
    previousBlockHash:{{K3ti0Agk7DEponywKcQCPRYVHb5RuyxdmQFTfrloptA=}},
    entriesHashList:[
      {{pbzvz6ofJC7mD2jvgfyrY/VtR01zIZHoWy8T1VcxlGo=}},
      {{k2brC23DLMercmiOWHiURaGwHu0mQtLzdNPuviE2rcs=}},
      {{hvw1EV8k4oOkIO36kblO/+UUSFUQqCanKuDGr0aP9nQ=}},
      {{ZrLbkyzDcpJ9KWsZMZqRuKUkG/czLIJ4US+K5E31b+Q=}}
    ],
    transactionInfo:{
      statements:[
        {
          statement:"SELECT * FROM Person WHERE GovId = ?",
          startTime:2019-09-18T17:00:14.587Z,
          statementDigest:{{p4Dn0DiuYD3Xm9UQQ75YLwmoMbSfJmopOmTfMnXs26M=}}
        },
        {
          statement:"INSERT INTO Person ?",
          startTime:2019-09-18T17:00:14.594Z,
          statementDigest:{{klMLkLfa5VJqk6JUPtHkQpOsDdG4HmuUaq/VaApQflU=}}
        },
        {
          statement:"INSERT INTO VehicleRegistration ?",
          startTime:2019-09-18T17:00:14.598Z,
          statementDigest:{{B0gO9BWVNrzRYFoe7t+GVLpJ6uZcLKf5t/chkfRhspI=}}
        }
      ],
      documents:{
        '7z2OpEBgVCvCtwvx4a2JGn':{
          tableName:"Person",
          tableId:"LSkFkQvkIOjCmpTZpkfpn9",
          statements:[1]
        },
        'K0FpsSLpydLDr7hi6KUzqk':{
          tableName:"VehicleRegistration",
          tableId:"Ad3A07z0ZffC7Gpso7BXyO",
          statements:[2]
        }
      }
```

```
    },
    revisionSummaries:[
      {
        hash:{{uDthuiqSy4FwjZssyCiyFd90XoPSlIwomHBdF/OrmkE=}},
        documentId:"7z2OpEBgVCvCtwvx4a2JGn"
      },
      {
        hash:{{qJID/amu0gN3dpG5Tg0FfIFTh/U5yFkfT+g/O6k5sPM=}},
        documentId:"K0FpsSLpydLDr7hi6KUzqk"
      }
    ]
  }
}
```

Sur le `revisionSummaries` terrain, certaines révisions peuvent ne pas comporter de`documentId`. Il s'agit de révisions internes du système qui ne contiennent aucune donnée utilisateur. Un flux QLDB inclut ces révisions dans leurs enregistrements récapitulatifs de blocs respectifs, car les hachages de ces révisions font partie de la chaîne de hachage complète de la revue. La chaîne de hachage complète est requise pour la vérification cryptographique.

Seules les révisions dotées d'un identifiant de document sont publiées dans des enregistrements de détails de révision distincts, comme décrit dans la section suivante.

Enregistrements détaillés des révisions

Un enregistrement détaillé des révisions représente une révision de document qui est enregistrée dans votre journal. La charge utile contient tous les attributs de la vue validée de la révision, ainsi que le nom et l'ID de table associés. Voici un exemple d'enregistrement de révision avec des exemples de données.

```
{
  qldbStreamArn:"arn:aws:qldb:us-east-1:123456789012:stream/exampleLedger/
IiPT4brpZCqCq3f4MTHbYy",
  recordType:"REVISION_DETAILS",
  payload:{
    tableInfo:{
      tableName:"VehicleRegistration",
      tableId:"Ad3A07z0ZffC7Gpso7BXyO"
    },
    revision:{
      blockAddress:{
        strandId:"ElYL30RGoqrFCbbaQn3K6m",
```

```
        sequenceNo:60807
      },
      hash:{{qJID/amu0gN3dpG5Tg0FfIFTh/U5yFkfT+g/O6k5sPM=}},
      data:{
        VIN:"1N4AL11D75C109151",
        LicensePlateNumber:"LEWISR261LL",
        State:"WA",
        City:"Seattle",
        PendingPenaltyTicketAmount:90.25,
        ValidFromDate:2017-08-21,
        ValidToDate:2020-05-11,
        Owners:{
          PrimaryOwner:{PersonId:"7z2OpEBgVCvCtwvx4a2JGn"},
          SecondaryOwners:[]
        }
      },
      metadata:{
        id:"K0FpsSLpydLDr7hi6KUzqk",
        version:0,
        txTime:2019-09-18T17:00:14.602Z,
        txId:"9RWohCo7My4GGkxRETAJ6M"
      }
    }
  }
}
```

Gestion des doublons et out-of-order des enregistrements

Les flux QLDB peuvent publier des doublons out-of-order et des enregistrements dans Kinesis Data Streams. Ainsi, une application grand public peut avoir besoin d'implémenter sa propre logique pour identifier et gérer de tels scénarios. Les enregistrements du résumé des blocs et des détails des révisions contiennent des champs que vous pouvez utiliser à cette fin. Associés aux fonctionnalités des services en aval, ces champs peuvent indiquer à la fois une identité unique et un ordre strict pour les enregistrements.

Prenons l'exemple d'un flux qui intègre QLDB à OpenSearch un index pour fournir des fonctionnalités de recherche en texte intégral sur des documents. Dans ce cas d'utilisation, vous devez éviter d'indexer les révisions obsolètes (out-of-order) d'un document. Pour appliquer le tri et la déduplication, vous pouvez utiliser les champs d'ID de document et de version externe dans OpenSearch, ainsi que les champs d'ID de document et de version dans un enregistrement détaillé des révisions.

Pour un exemple de logique de déduplication dans un exemple d'application intégrant QLDB à OpenSearch Amazon Service, GitHub consultez le référentiel aws-samples/ -. amazon-qldb-streaming-amazon opensearch-service-sample-python

Autorisations de diffusion dans QLDB

Avant de créer un flux Amazon QLDB, vous devez fournir à QLDB des autorisations d'écriture sur la ressource Amazon Kinesis Data Streams que vous avez spécifiée. Si vous utilisez un client géré AWS KMS key pour le chiffrement côté serveur de votre flux Kinesis, vous devez également autoriser QLDB à utiliser la clé de chiffrement symétrique que vous avez spécifiée. Kinesis Data Streams ne prend pas en charge les clés KMS asymétriques.

Pour fournir à votre flux QLDB les autorisations nécessaires, vous pouvez faire en sorte que QLDB assume un rôle de service IAM avec les politiques d'autorisation appropriées. Une fonction de service est un rôle IAM qu'un service endosse pour accomplir des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer une fonction du service à partir d'IAM. Pour plus d'informations, consultez Création d'un rôle pour la délégation d'autorisations à un Service AWS dans le Guide de l'utilisateur IAM.

> **Note**
>
> Pour transmettre un rôle à QLDB lorsque vous demandez un flux de journal, vous devez disposer des autorisations nécessaires pour effectuer l'action `iam:PassRole` sur la ressource de rôle IAM. Cela s'ajoute à l'`qldb:StreamJournalToKinesis`autorisation sur la sous-ressource de flux QLDB.
> Pour savoir comment contrôler l'accès à QLDB à l'aide d'IAM, consultez. Comment Amazon QLDB fonctionne avec IAM Pour un exemple de politique QLDB, consultez. Exemples de politiques basées sur l'identité pour Amazon QLDB

Dans cet exemple, vous créez un rôle qui permet à QLDB d'écrire des enregistrements de données dans un flux de données Kinesis en votre nom. Pour plus d'informations, consultez Création d'un rôle pour la délégation d'autorisations à un Service AWS dans le Guide de l'utilisateur IAM.

Si vous diffusez un journal QLDB dans Compte AWS votre journal pour la première fois, vous devez d'abord créer un rôle IAM avec les politiques appropriées en procédant comme suit. Vous pouvez également utiliser la console QLDB pour créer automatiquement le rôle pour vous. Sinon, vous pouvez choisir un rôle que vous avez créé précédemment.

Rubriques

Créer une stratégie d’autorisations

Procédez comme suit pour créer une politique d'autorisation pour un flux QLDB. Cet exemple montre une politique Kinesis Data Streams qui accorde à QLDB l'autorisation d'écrire des enregistrements de données dans le flux de données Kinesis que vous avez spécifié. Le cas échéant, l'exemple montre également une politique de clé qui permet à QLDB d'utiliser votre clé KMS de chiffrement symétrique.

Pour plus d'informations sur les politiques de Kinesis Data Streams, consultez la section Contrôle de l'accès aux ressources Amazon Kinesis Data Streams à l'aide de l'IAM et des autorisations pour utiliser les clés KMS générées par les utilisateurs dans le manuel Amazon Kinesis Data Streams Developer Guide. Pour en savoir plus sur les politiques AWS KMS clés, consultez la section Utilisation des politiques clés AWS KMS dans le Guide du AWS Key Management Service développeur.

> **Note**
>
> Votre flux de données Kinesis et votre clé KMS doivent tous deux se trouver dans le même Région AWS compte que votre registre QLDB.

Pour utiliser l'éditeur de politique JSON afin de créer une politique

1. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/iam/.
2. Dans la colonne de navigation de gauche, sélectionnez Politiques.

   Si vous choisissez Politiques pour la première fois, la page Bienvenue dans les politiques gérées s'affiche. Sélectionnez Mise en route.
3. En haut de la page, sélectionnez Créer une politique.
4. Sélectionnez l’onglet JSON.
5. Entrez un document de stratégie JSON.
   - Si vous utilisez une clé KMS gérée par le client pour le chiffrement côté serveur de votre flux Kinesis, utilisez l'exemple de document de politique suivant. *Pour*

utiliser cette politique, remplacez us-east-1, 123456789012 et 1234abcd-12ab-34cd-56ef-1234567890ab dans l'exemple par vos propres kinesis-stream-nameinformations.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBStreamKinesisPermissions",
            "Action": [ "kinesis:PutRecord*", "kinesis:DescribeStream",
 "kinesis:ListShards" ],
            "Effect": "Allow",
            "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/kinesis-
stream-name"
        },
        {
            "Sid": "QLDBStreamKMSPermission",
            "Action": [ "kms:GenerateDataKey" ],
            "Effect": "Allow",
            "Resource": "arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ]
}
```

- Sinon, utilisez l'exemple de document de politique suivant. Pour utiliser cette politique, remplacez *us-east-1*, 123456789012*, et* dans l'exemple par vos propres informations. *kinesis-stream-name*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBStreamKinesisPermissions",
            "Action": [ "kinesis:PutRecord*", "kinesis:DescribeStream",
 "kinesis:ListShards" ],
            "Effect": "Allow",
            "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/kinesis-
stream-name"
        }
    ]
}
```

6. Choisissez Examiner une politique.

> **Note**
>
> Vous pouvez basculer à tout moment entre les onglets Éditeur visuel et JSON. Toutefois, si vous apportez des modifications ou sélectionnez Examiner une politique dans l'onglet Éditeur visuel, IAM peut restructurer votre politique pour optimiser son affichage dans l'éditeur visuel. Pour de plus amples informations, consulte Restructuration d'une politique dans le Guide de l'utilisateur IAM.

7. Dans la page Examiner une politique, entrez un nom et éventuellement une description pour la politique que vous êtes en train de créer. Vérifiez le récapitulatif de la politique pour voir les autorisations accordées par votre politique. Sélectionnez ensuite Créer une politique pour enregistrer votre travail.

Créer un rôle IAM

Après avoir créé une politique d'autorisation pour votre flux QLDB, vous pouvez créer un rôle IAM et y associer votre politique.

Pour créer le rôle de service pour QLDB (console IAM)

1. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation de la console IAM, sélectionnez Roles (Rôles), puis Create role (Créer un rôle).
3. Pour Trusted entity (Entité de confiance), choisissez Service AWS.
4. Pour Service ou cas d'utilisation, choisissez QLDB, puis choisissez le cas d'utilisation QLDB.
5. Choisissez Suivant.
6. Cochez la case à côté de la politique que vous avez créée lors des étapes précédentes.
7. (Facultatif) Définissez une limite d'autorisations. Il s’agit d’une fonctionnalité avancée disponible pour les fonctions de service, mais pas pour les rôles liés à un service.

   a. Ouvrez la section Définir les limites des autorisations, puis choisissez Utiliser une limite d'autorisations pour contrôler le nombre maximal d'autorisations de rôle.

      IAM inclut une liste des politiques AWS gérées et gérées par le client dans votre compte.

b. Sélectionnez la politique à utiliser comme limite d'autorisations.

8. Choisissez Suivant.
9. Entrez un nom de rôle ou un suffixe de nom de rôle pour vous aider à identifier l'objectif du rôle.

> **Important**
>
> Lorsque vous nommez un rôle, tenez compte des points suivants :
>
> - Les noms de rôles doivent être uniques au sein du Compte AWS vôtre et ne peuvent pas être rendus uniques au cas par cas.
>
> Par exemple, ne créez pas de rôles nommés à la fois **`PRODROLE`** et**`prodrole`**. Lorsqu'un nom de rôle est utilisé dans une politique ou dans le cadre d'un ARN, il distingue les majuscules et minuscules, mais lorsqu'un nom de rôle apparaît aux clients dans la console, par exemple pendant le processus de connexion, le nom du rôle ne fait pas la distinction entre majuscules et minuscules.
> - Vous ne pouvez pas modifier le nom du rôle une fois qu'il a été créé car d'autres entités peuvent y faire référence.

10. (Facultatif) Dans Description, entrez une description pour le rôle.
11. (Facultatif) Pour modifier les cas d'utilisation et les autorisations du rôle, dans les sections Étape 1 : Sélection des entités de confiance ou Étape 2 : Ajouter des autorisations, choisissez Modifier.
12. (Facultatif) Pour identifier, organiser ou rechercher le rôle, ajoutez des balises sous forme de paires clé-valeur. Pour plus d'informations sur l'utilisation des balises dans IAM, consultez la rubrique Balisage des ressources IAM dans le Guide de l'utilisateur IAM.
13. Passez en revue les informations du rôle, puis choisissez Create role (Créer un rôle).

Le document JSON suivant est un exemple de politique de confiance qui permet à QLDB d'assumer un rôle IAM auquel des autorisations spécifiques sont associées.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "qldb.amazonaws.com"
```

```
            },
            "Action": [ "sts:AssumeRole" ],
            "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:qldb:us-
east-1:123456789012:stream/myExampleLedger/*"
                },
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                }
            }
        }
    ]
}
```

> **Note**
>
> Cet exemple de politique de confiance montre comment vous pouvez utiliser les touches `aws:SourceArn` contextuelles et de condition `aws:SourceAccount` globale pour éviter le problème de confusion des adjoints. Grâce à cette politique de confiance, QLDB peut assumer le rôle de n'importe quel flux QLDB dans le compte uniquement pour le registre. `123456789012` `myExampleLedger`
> Pour plus d'informations, consultez Prévention du cas de figure de l'adjoint désorienté entre services.

Après avoir créé votre rôle IAM, revenez à la console QLDB et actualisez la page Créer un flux QLDB afin qu'elle puisse trouver votre nouveau rôle.

Erreurs courantes pour les flux de journaux dans QLDB

Cette section décrit les erreurs d'exécution générées par Amazon QLDB pour les requêtes de flux de journaux.

Voici une liste des exceptions courantes renvoyées par le service. Chaque exception inclut un message d'erreur spécifique, suivi d'une brève description et de suggestions de solutions possibles.

AccessDeniedException

Message : Utilisateur : UserArn n'est pas autorisé à exécuter : iam : PassRole on resource : Rolearn

Vous n'êtes pas autorisé à transmettre un rôle IAM au service QLDB. QLDB nécessite un rôle pour toutes les demandes de flux de journal, et vous devez disposer des autorisations nécessaires pour transmettre ce rôle à QLDB. Le rôle fournit à QLDB des autorisations d'écriture dans la ressource Amazon Kinesis Data Streams que vous avez spécifiée.

Vérifiez que vous définissez une politique IAM qui autorise l'exécution de l'opération d'`PassRole`API sur la ressource de rôle IAM que vous avez spécifiée pour le service QLDB (). `qldb.amazonaws.com` Pour un exemple de stratégie, consultez Exemples de politiques basées sur l'identité pour Amazon QLDB.

IllegalArgumentException

Message : QLDB a rencontré une erreur lors de la validation de Kinesis Data Streams : Response from Kinesis : ErrorCode ErrorMessage

Cette erreur peut être due au fait que la ressource Kinesis Data Streams fournie n'existe pas. Ou bien, QLDB ne dispose pas des autorisations suffisantes pour écrire des enregistrements de données dans le flux de données Kinesis que vous avez spécifié.

Vérifiez que le flux de données Kinesis que vous fournissez dans votre demande de flux est correct. Pour plus d'informations, consultez la section Création et mise à jour de flux de données dans le manuel Amazon Kinesis Data Streams Developer Guide.

Vérifiez également que vous définissez une politique pour le flux de données Kinesis que vous avez spécifié qui accorde au service QLDB (`qldb.amazonaws.com`) des autorisations pour les actions suivantes. Pour plus d'informations, consultez Autorisations de diffusion.

- `kinesis:PutRecord`
- `kinesis:PutRecords`
- `kinesis:DescribeStream`
- `kinesis:ListShards`

IllegalArgumentException

Message : Réponse inattendue de Kinesis Data Streams lors de la validation de la configuration Kinesis. *Réponse de Kinesis : ErrorCode ErrorMessage*

La tentative d'écriture d'enregistrements de données dans le flux de données Kinesis fourni a échoué avec la réponse d'erreur Kinesis fournie. Pour plus d'informations sur les causes possibles, consultez la section Résolution des problèmes liés aux producteurs d'Amazon Kinesis Data Streams dans le manuel du développeur Amazon Kinesis Data Streams.

IllegalArgumentException

Message : La date de début ne doit pas être supérieure à la date de fin.

Les deux `InclusiveStartTime` `ExclusiveEndTime` doivent être au format de date et d'heure ISO 8601 et en temps universel coordonné (UTC).

IllegalArgumentException

Message : La date de début ne peut pas être future.

Les deux `InclusiveStartTime` `ExclusiveEndTime` doivent être au format `ISO 8601` date et heure et en UTC.

LimitExceededException

Message : Dépassement de la limite de 5 flux de Journal exécutés simultanément vers Kinesis Data Streams

QLDB impose une limite par défaut de cinq flux de journaux simultanés.

Gestion des livres dans Amazon QLDB

Ce chapitre explique comment utiliser l'API QLDB, laAWS Command Line Interface (AWS CLI), et commentAWS CloudFormation effectuer des opérations de gestion de livres dans Amazon QLDB.

Vous pouvez également effectuer ces tâches à partir d'AWS Management Console. Pour plus d'informations, consultez Accès à Amazon QLDB à l'aide de la console.

Rubriques

- Opérations de base pour les registres Amazon QLDB
- Création de ressources Amazon QLDB avecAWS CloudFormation
- Balisage des ressources Amazon QLDB

Opérations de base pour les registres Amazon QLDB

Vous pouvez utiliser l'API QLDB ouAWS Command Line Interface (AWS CLI) pour créer, mettre à jour et supprimer des livres dans Amazon QLDB. Vous pouvez également répertorier tous les registres de votre compte ou obtenir des informations sur un registre spécifique.

Les rubriques suivantes fournissent de courts exemples de code qui montrent les étapes courantes pour les opérations de registre utilisant leAWS SDK for Java et leAWS CLI.

Rubriques

- Création d'un registre
- Description d'un registre
- Mettre à jour un registre
- Mettre à jour le mode d'autorisations d'un registre
- Supprimer un registre
- Listes de listage

Pour obtenir des exemples de code illustrant ces opérations dans un exemple d'application complet, consultez lesDémarrage didacticiels et GitHub référentiels suivants :

- Java : Tutoriel | GitHub référentiel

- Node.js : Tutoriel | GitHub référentiel
- Python : Tutoriel | GitHub référentiel

Création d'un registre

Utilisez cette`CreateLedger` opération pour créer un registre dans votreCompte AWS. Vous devez fournir les informations suivantes :

- Nom du registre — Nom du registre que vous souhaitez créer sur votre compte. Le nom doit être unique parmi tous vos registres actuelsRégion AWS.

  Les contraintes de dénomination pour les noms de registre sont définies dansQuotas et limites d'Amazon QLDB.
- Mode d'autorisations — Mode d'autorisations à attribuer au registre. Choisissez l'une des options suivantes :
  - Tout autoriser — un mode d'autorisations hérité qui permet le contrôle d'accès avec une granularité au niveau de l'API pour les registres.

    Ce mode permet aux utilisateurs qui possèdent l'autorisation d'API `SendCommand` pour ce registre d'exécuter toutes les commandes PartiQL (par conséquent, `ALLOW_ALL`) sur toutes les tables du registre spécifié. Ce mode ignore les politiques d'autorisations IAM au niveau de la table ou de la commande que vous créez pour le registre.
  - Standard — (Recommandé) un mode d'autorisations qui permet le contrôle d'accès avec une granularité plus fine pour les registres, les tables et les commandes PartiQL. Nous vous recommandons vivement d'utiliser ce mode d'autorisations pour optimiser la sécurité des données de votre registre.

    Par défaut, ce mode refuse toutes les demandes d'exécuter des commandes PartiQL sur les tables de ce registre. Pour autoriser les commandes PartiQL, vous devez créer des politiques d'autorisations IAM pour des ressources de table spécifiques et des actions PartiQL, en plus de l'autorisation d'`SendCommand`API pour le registre. Pour plus d'informations, consultez Commencer à utiliser le mode d'autorisation standard dans Amazon QLDB.
- Protection contre la suppression — (Facultatif) Indicateur qui empêche la suppression d'un registre par n'importe quel utilisateur. Si vous ne la spécifiez pas lors de la création du registre, cette fonction est activée (`true`) par défaut.

Si la protection contre la suppression est activée, vous devez commencer par la désactiver avant de pouvoir supprimer le registre. Vous pouvez la désactiver en utilisant l'`UpdateLedger`opération pour définir l'indicateur sur`false`.

- AWS KMS key— (Facultatif) La clé dansAWS Key Management Service (AWS KMS) à utiliser pour le chiffrement des données au repos. Choisissez l'un des types suivants deAWS KMS keys :
  - AWSclé KMS détenue et gérée par en votreAWS nom.

    Si vous ne définissez pas ce paramètre lors de la création du registre, le registre utilise ce type de clé par défaut. Vous pouvez également utiliser la chaîne`AWS_OWNED_KMS_KEY` pour spécifier ce type de clé. Cette option ne nécessite aucune configuration supplémentaire.
  - Clé KMS gérée par le client — Utilisez une clé KMS de chiffrement symétrique dans votre compte que vous créez, possédez et gérez. QLDB ne prend pas en charge les touches asymétriques.

    Cette option vous oblige à créer une clé KMS ou à utiliser une clé existante dans votre compte. Pour obtenir des instructions sur la création d'une clé gérée par le client, consultez la section Création de clés KMS de chiffrement symétriques dans le Guide duAWS Key Management Service développeur.

    Vous pouvez spécifier une clé KMS gérée par le client à l'aide d'un ID, d'un alias ou d'Amazon Resource Name (ARN). Pour en savoir plus, consultez la section Identifiants clés (KeyId) dans le Guide duAWS Key Management Service développeur.

    > **Note**
    >
    > Les clés inter-régions ne sont pas prises en charge. La clé KMS spécifiée doit se trouver dans le même registreRégion AWS que votre registre.

  Pour plus d'informations, consultez Chiffrement au repos dans Amazon QLDB.
- Etiquettes — (Facultatif) Ajoutez des métadonnées au registre en associant les balises sous forme de paires clé-valeur. Vous pouvez ajouter des balises à votre registre pour mieux les organiser et les identifier. Pour plus d'informations, consultez Balisage des ressources Amazon QLDB.

Le registre n'est pas prêt à être utilisé jusqu'à ce que QLDB le crée et définisse son état sur`ACTIVE`.

Création d'un registre (Java)

Pour créer un registre à l'aide duAWS SDK for Java

1. Créez une instance de la classe `AmazonQLDB`.
2. Créez une instance de la classe `CreateLedgerRequest` pour fournir l'information de requête.

   Vous devez fournir le nom du registre et un mode d'autorisation.
3. Exécutez la méthode `createLedger` en fournissant l'objet de demande comme paramètre.

La`createLedger` demande renvoie un`CreateLedgerResult` objet contenant des informations sur le registre. Reportez-vous à la section suivante pour voir un exemple d'utilisation de l'`DescribeLedger`opération pour vérifier l'état de votre registre après sa création.

Les exemples suivants illustrent les étapes précédentes.

Example — Utiliser les paramètres de configuration par défaut

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
CreateLedgerRequest request = new CreateLedgerRequest()
        .withName(ledgerName)
        .withPermissionsMode(PermissionsMode.STANDARD);
CreateLedgerResult result = client.createLedger(request);
```

> **Note**
>
> Le registre utilise les paramètres par défaut suivants si vous ne les spécifiez pas :
>
> - Protection contre la suppression : activée (`true`).
> - Clé KMS : clé KMSAWS détenue.

Example — Désactivez la protection contre la suppression, utilisez une clé KMS gérée par le client et attachez des balises

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();

Map<String, String> tags = new HashMap<>();
tags.put("IsTest", "true");
```

```
tags.put("Domain", "Test");

CreateLedgerRequest request = new CreateLedgerRequest()
        .withName(ledgerName)
        .withPermissionsMode(PermissionsMode.STANDARD)
        .withDeletionProtection(false)
        .withKmsKey("arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab")
        .withTags(tags);
CreateLedgerResult result = client.createLedger(request);
```

Création d'un registre (AWS CLI)

Créez un nouveau registre nommé à`vehicle-registration` l'aide des paramètres de configuration par défaut.

Example

```
aws qldb create-ledger --name vehicle-registration --permissions-mode STANDARD
```

> **Note**
>
> Le registre utilise les paramètres par défaut suivants si vous ne les spécifiez pas :
>
> - Protection contre la suppression : activée (`true`).
> - Clé KMS : clé KMSAWS détenue.

Vous pouvez également créer un nouveau registre nommé`vehicle-registration` avec la protection contre la suppression désactivée, avec une clé KMS spécifique gérée par le client et des balises spécifiées.

Example

```
aws qldb create-ledger \
    --name vehicle-registration \
    --no-deletion-protection \
    --permissions-mode STANDARD \
    --kms-key arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab \
    --tags IsTest=true,Domain=Test
```

Création d'un registre (AWS CloudFormation)

Vous pouvez également utiliser un AWS CloudFormationmodèle pour créer des livres. Pour plus d'informations, consultez la AWS::QLDB::Ledgerressource dans le Guide deAWS CloudFormation l'utilisateur.

Description d'un registre

Utilisez l'`DescribeLedger`opération pour afficher les détails d'un registre. Vous devez fournir le nom du registre. La sortie de `DescribeLedger` est au même format que celle de `CreateLedger`. Elle contient les informations suivantes :

- Nom du registre — Nom du registre que vous souhaitez décrire.
- ARN — Amazon Resource Name (ARN) pour le registre au format suivant.

```
arn:aws:qldb:aws-region:account-id:ledger/ledger-name
```

- Protection contre la suppression : indicateur indiquant si la fonction de protection contre la suppression est activée.
- Date et heure de création — Date et heure, au format d'époque, auxquelles le registre a été créé.
- État : statut actuel du registre. Il peut s'agir de l'une des valeurs suivantes :
  - `CREATING`
  - `ACTIVE`
  - `DELETING`
  - `DELETED`
- Mode autorisations : mode d'autorisations attribué au registre. Il peut s'agir de l'une des valeurs suivantes :
  - `ALLOW_ALL`— un mode d'autorisations hérité qui permet le contrôle d'accès avec une granularité au niveau de l'API pour les registres.
  - `STANDARD`— un mode d'autorisations qui permet le contrôle d'accès avec une granularité plus fine pour les registres, les tables et les commandes PartiQL.
- Description du chiffrement — Informations sur le chiffrement des données au repos dans le registre. Cela inclut les éléments suivants :
  - AWS KMS keyARN : ARN de la clé KMS gérée par le client que le registre utilise pour le chiffrement au repos. Si elle n'est pas définie, le registre utilise une clé KMSAWS détenue pour le chiffrement.

- État du chiffrement : état actuel du chiffrement au repos pour le registre. Il peut s'agir de l'une des valeurs suivantes :
  - `ENABLED`— Le chiffrement est entièrement activé à l'aide de la clé spécifiée.
  - `UPDATING`— Le changement de clé spécifié est en cours de traitement.

    Les principales modifications apportées à QLDB sont asynchrones. Le registre est entièrement accessible sans aucun impact sur les performances pendant le traitement de la modification clé. Le temps nécessaire à la mise à jour d'une clé varie en fonction de la taille du registre.
  - `KMS_KEY_INACCESSIBLE`— La clé KMS gérée par le client spécifiée n'est pas accessible et le registre est endommagé. Soit la clé a été désactivée, soit supprimée, soit les autorisations associées à la clé ont été révoquées. Lorsqu'un registre est endommagé, il n'est pas accessible et n'accepte aucune demande de lecture ou d'écriture.

    Un registre défectueux revient automatiquement à son état actif une fois que vous avez rétabli les autorisations accordées à la clé ou après avoir réactivé la clé désactivée. Toutefois, la suppression d'une clé KMS gérée par le client est irréversible. Après la suppression d'une clé, vous ne pouvez plus accéder aux registres protégés par cette clé et les données deviennent irrémédiables de façon permanente.
- InaccessibleAWS KMS key : date et heure, au format epoch time, auxquelles la clé KMS est devenue inaccessible pour la première fois, en cas d'erreur.

  Ce paramètre n'est pas défini si la clé KMS est accessible.

Pour plus d'informations, consultez Chiffrement au repos dans Amazon QLDB.

> **Note**
>
> Une fois que vous avez créé un registre QLDB, il est prêt à être utilisé lorsque son statut passe de`CREATING` à`ACTIVE`.

Décrire un registre (Java)

Pour décrire un registre à l'aide duAWS SDK for Java

1. Créez une instance de la classe `AmazonQLDB`. Vous pouvez également utiliser la même instance du`AmazonQLDB` client que vous avez instanciée pour la`CreateLedger` demande.

2. Créez une instance de la`DescribeLedgerRequest` classe et spécifiez le nom de registre que vous souhaitez décrire.
3. Exécutez la méthode `describeLedger` en fournissant l'objet de demande comme paramètre.
4. La`describeLedger` demande renvoie un`DescribeLedgerResult` objet contenant des informations actuelles sur le registre.

L'exemple de code suivant illustre les étapes précédentes. Vous pouvez appeler la`describeLedger` méthode du client pour obtenir des informations sur le registre à tout moment.

Example

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
DescribeLedgerRequest request = new DescribeLedgerRequest().withName(ledgerName);
DescribeLedgerResult result = client.describeLedger(request);
System.out.printf("%s: ARN: %s \t State: %s \t CreationDateTime: %s \t
 DeletionProtection: %s
                   \t PermissionsMode: %s \t EncryptionDescription: %s",
        result.getName(),
        result.getArn(),
        result.getState(),
        result.getCreationDateTime(),
        result.getDeletionProtection(),
        result.getPermissionsMode(),
        result.getEncryptionDescription());
```

Décrire un registre (AWS CLI)

Décrivez le`vehicle-registration` registre que vous venez de créer.

Example

```
aws qldb describe-ledger --name vehicle-registration
```

Mettre à jour un registre

L'`UpdateLedger`opération vous permet actuellement de modifier les paramètres de configuration suivants pour un registre existant :

- Protection contre la suppression — Indicateur qui empêche la suppression d'un registre par n'importe quel utilisateur. Si cette fonction est activée, vous devez commencer par la désactiver en définissant l'indicateur sur`false` avant de pouvoir supprimer le registre.

  Si vous ne définissez pas ce paramètre, aucune modification n'est apportée au paramètre de protection contre la suppression du registre.
- AWS KMS key— La cléAWS Key Management Service (AWS KMS) à utiliser pour le chiffrement des données au repos. Si vous ne définissez pas ce paramètre, aucune modification n'est apportée à la clé KMS du registre.

  > **Note**
  >
  > Amazon QLDB a lancé le support pour les clients gérésAWS KMS keys le 22 juillet 2021. Tous les registres créés avant le lancement sont protégésClés détenues par AWS par défaut, mais ne peuvent actuellement pas être chiffrés au repos à l'aide de clés gérées par le client.
  > Vous pouvez consulter l'heure de création de votre registre sur la console QLDB.

  Utilisez l'une des options suivantes :

  - AWSclé KMS détenue et gérée par en votreAWS nom. Pour utiliser ce type de clé, spécifiez la chaîne`AWS_OWNED_KMS_KEY` pour ce paramètre. Cette option ne nécessite aucune configuration supplémentaire.
  - Clé KMS gérée par le client — Utilisez une clé KMS de chiffrement symétrique dans votre compte que vous créez, possédez et gérez. QLDB ne prend pas en charge les touches asymétriques.

    Cette option vous oblige à créer une clé KMS ou à utiliser une clé existante dans votre compte. Pour obtenir des instructions sur la création d'une clé gérée par le client, consultez la section Création de clés KMS de chiffrement symétriques dans le Guide duAWS Key Management Service développeur.

    Vous pouvez spécifier une clé KMS gérée par le client à l'aide d'un ID, d'un alias ou d'Amazon Resource Name (ARN). Pour en savoir plus, consultez la section Identifiants clés (KeyId) dans le Guide duAWS Key Management Service développeur.

> **Note**
>
> Les clés inter-régions ne sont pas prises en charge. La clé KMS spécifiée doit se trouver dans le même registreRégion AWS que votre registre.

Les principales modifications apportées à QLDB sont asynchrones. Le registre est entièrement accessible sans aucun impact sur les performances pendant le traitement de la modification clé.

Vous pouvez changer de clé aussi souvent que nécessaire, mais le temps nécessaire à la mise à jour d'une clé varie en fonction de la taille du registre. Vous pouvez utiliser cette`DescribeLedger` opération pour vérifier l'état du cryptage au repos.

Pour plus d'informations, consultez Chiffrement au repos dans Amazon QLDB.

La sortie de `UpdateLedger` est au même format que celle de `CreateLedger`.

Mettre à jour un registre (Java)

Pour mettre à jour un registre à l'aide duAWS SDK for Java

1. Créez une instance de la classe `AmazonQLDB`.
2. Créez une instance de la classe `UpdateLedgerRequest` pour fournir l'information de requête.

   Vous devez fournir le nom du livre ainsi qu'une nouvelle valeur booléenne pour la protection contre la suppression ou une nouvelle valeur de chaîne pour la clé KMS.
3. Exécutez la méthode `updateLedger` en fournissant l'objet de demande comme paramètre.

Les exemples de code suivants illustrent les étapes précédentes. Lau`pdateLedger` demande renvoie un`UpdateLedgerResult` objet contenant des informations mises à jour sur le registre.

Example — Désactiver la protection contre la suppression

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
UpdateLedgerRequest request = new UpdateLedgerRequest()
        .withName(ledgerName)
        .withDeletionProtection(false);
UpdateLedgerResult result = client.updateLedger(request);
```

Example — Utiliser une clé KMS gérée par le client

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
UpdateLedgerRequest request = new UpdateLedgerRequest()
        .withName(ledgerName)
        .withKmsKey("arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab")
UpdateLedgerResult result = client.updateLedger(request);
```

Example — Utilisez une clé KMSAWS détenue

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
UpdateLedgerRequest request = new UpdateLedgerRequest()
        .withName(ledgerName)
        .withKmsKey("AWS_OWNED_KMS_KEY")
UpdateLedgerResult result = client.updateLedger(request);
```

Mettre à jour un registre (AWS CLI)

Si la protection contre la suppression est activée sur votre`vehicle-registration` registre, vous devez commencer par la désactiver avant de pouvoir la supprimer.

Example

```
aws qldb update-ledger --name vehicle-registration --no-deletion-protection
```

Vous pouvez également modifier les paramètres de chiffrement au repos du registre pour utiliser une clé KMS gérée par le client.

Example

```
aws qldb update-ledger --name vehicle-registration --kms-key arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

Vous pouvez également modifier les paramètres de chiffrement au repos pour utiliser une clé KMSAWS détenue.

Example

```
aws qldb update-ledger --name vehicle-registration --kms-key AWS_OWNED_KMS_KEY
```

Mettre à jour le mode d'autorisations d'un registre

L'`UpdateLedgerPermissionsMode`opération vous permet de modifier le mode d'autorisation d'un registre existant. Choisissez l'une des options suivantes :

- Tout autoriser — un mode d'autorisations hérité qui permet le contrôle d'accès avec une granularité au niveau de l'API pour les registres.

  Ce mode permet aux utilisateurs qui possèdent l'autorisation d'API `SendCommand` pour ce registre d'exécuter toutes les commandes PartiQL (par conséquent, `ALLOW_ALL`) sur toutes les tables du registre spécifié. Ce mode ignore les politiques d'autorisations IAM au niveau de la table ou de la commande que vous créez pour le registre.
- Standard — (Recommandé) un mode d'autorisations qui permet le contrôle d'accès avec une granularité plus fine pour les registres, les tables et les commandes PartiQL. Nous vous recommandons vivement d'utiliser ce mode d'autorisations pour optimiser la sécurité des données de votre registre.

  Par défaut, ce mode refuse toutes les demandes d'exécuter des commandes PartiQL sur les tables de ce registre. Pour autoriser les commandes PartiQL, vous devez créer des politiques d'autorisations IAM pour des ressources de table spécifiques et des actions PartiQL, en plus de l'autorisation d'`SendCommand`API pour le registre. Pour plus d'informations, consultez Commencer à utiliser le mode d'autorisation standard dans Amazon QLDB.

> **Important**
>
> Avant de passer en mode`STANDARD` autorisations, vous devez d'abord créer toutes les politiques IAM et les balises de table requises afin d'éviter toute interruption pour vos utilisateurs. Pour en savoir plus, passez àMigration vers le mode d'autorisations standard.

Mettre à jour le mode d'autorisations d'un registre (Java)

Pour mettre à jour le mode d'autorisations d'un registre à l'aide duAWS SDK for Java

1. Créez une instance de la classe `AmazonQLDB`.
2. Créez une instance de la classe `UpdateLedgerPermissionsModeRequest` pour fournir l'information de requête.

Vous devez fournir le nom du livre ainsi qu'une nouvelle valeur de chaîne pour le mode d'autorisations.

3. Exécutez la méthode `updateLedgerPermissionsMode` en fournissant l'objet de demande comme paramètre.

Les exemples de code suivants illustrent les étapes précédentes. La`updateLedgerPermissionsMode` demande renvoie un`UpdateLedgerPermissionsModeResult` objet contenant des informations mises à jour sur le registre.

Example — Attribuez le mode d'autorisation standard

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
UpdateLedgerPermissionsModeRequest request = new UpdateLedgerPermissionsModeRequest()
        .withName(ledgerName)
        .withPermissionsMode(PermissionsMode.STANDARD);
UpdateLedgerPermissionsModeResult result = client.updateLedgerPermissionsMode(request);
```

Mettre à jour le mode d'autorisations d'un registre (AWS CLI)

Attribuez le mode d'`STANDARD`autorisations à votre`vehicle-registration` registre.

Example

```
aws qldb update-ledger-permissions-mode --name vehicle-registration --permissions-mode
 STANDARD
```

Migration vers le mode d'autorisations standard

Pour passer au mode`STANDARD` autorisations, nous vous recommandons d'analyser vos modèles d'accès QLDB et d'ajouter des politiques IAM qui accordent à vos utilisateurs les autorisations appropriées pour accéder à leurs ressources.

Avant de passer en mode`STANDARD` autorisations, vous devez d'abord créer toutes les politiques IAM et les balises de table requises. Dans le cas contraire, le fait de changer de mode d'autorisation peut perturber les utilisateurs jusqu'à ce que vous créiez les politiques IAM appropriées ou que vous ne reveniez pas au mode d'autorisation`ALLOW_ALL`. Pour plus d'informations sur la création de telles politiques, consultezCommencer à utiliser le mode d'autorisation standard dans Amazon QLDB.

Vous pouvez également utiliser une politiqueAWS gérée pour accorder un accès complet à toutes les ressources QLDB. Les politiques`AmazonQLDBConsoleFullAccess` gérées`AmazonQLDBFullAccess` et incluent toutes les actions QLDB, y compris toutes les actions partiQL. Le fait d'associer l'une de ces politiques à un principal équivaut au mode`ALLOW_ALL` d'autorisation pour ce principal. Pour plus d'informations, consultez AWS politiques gérées pour Amazon QLDB.

Supprimer un registre

Utilisez cette`DeleteLedger` opération pour supprimer un registre et tout son contenu. La suppression d'un registre est une opération irrécupérable.

Si la protection contre la suppression est activée pour votre registre, vous devez commencer par la désactiver avant de pouvoir la supprimer.

Lorsque vous émettez une`DeleteLedger` demande, l'état du registre passe de`ACTIVE` à`DELETING`. La suppression du registre peut prendre un certain temps, en fonction de la quantité de stockage utilisée. Une fois l'`DeleteLedger`opération terminée, le registre n'existe plus dans QLDB.

Supprimer un registre (Java)

Pour supprimer un registre à l'aide duAWS SDK for Java

1. Créez une instance de la classe `AmazonQLDB`.
2. Créez une instance de la`DeleteLedgerRequest` classe et spécifiez le nom du registre que vous souhaitez supprimer.
3. Exécutez la méthode `deleteLedger` en fournissant l'objet de demande comme paramètre.

L'exemple de code suivant illustre les étapes précédentes.

Example

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
DeleteLedgerRequest request = new DeleteLedgerRequest().withName(ledgerName);
DeleteLedgerResult result = client.deleteLedger(request);
```

Supprimer un registre (AWS CLI)

Supprimez votre`vehicle-registration` registre.

Example

```
aws qldb delete-ledger --name vehicle-registration
```

Listes de listage

L'`ListLedgers`opération renvoie des informations récapitulatives de tous les livres QLDB pour le journal actuelCompte AWS et la région.

Ledgers de listage (Java)

Pour répertorier les livres de votre compte à l'aide duAWS SDK for Java

1. Créez une instance de la classe `AmazonQLDB`.
2. Créez une instance de la classe `ListLedgersRequest`.

   Si vous avez reçu une valeur pour`NextToken` dans la réponse à un`ListLedgers` appel précédent, vous devez fournir cette valeur dans cette demande pour obtenir la page de résultats suivante.
3. Exécutez la méthode `listLedgers` en fournissant l'objet de demande comme paramètre.
4. La`listLedgers` demande renvoie un`ListLedgersResult` objet. Cet objet possède une liste d'`LedgerSummary`objets et un jeton de pagination qui indique si d'autres résultats sont disponibles :

   - Si elle`NextToken` est vide, cela signifie que la dernière page de résultats a été traitée et qu'il n'y a plus de résultats.
   - S'il n'`NextToken`est pas vide, d'autres résultats sont disponibles. Pour récupérer la page de résultats suivante, utilisez la valeur de`NextToken` lors d'un`ListLedgers` appel suivant.

L'exemple de code suivant illustre les étapes précédentes.

Example

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
List<LedgerSummary> ledgerSummaries = new ArrayList<>();
String nextToken = null;
do {
    ListLedgersRequest request = new ListLedgersRequest().withNextToken(nextToken);
    ListLedgersResult result = client.listLedgers(request);
```

```
    ledgerSummaries.addAll(result.getLedgers());
    nextToken = result.getNextToken();
} while (nextToken != null);
```

Ledgers de cotation (AWS CLI)

Répertoriez tous les livres de la régionCompte AWS et de la région en cours.

Example

```
aws qldb list-ledgers
```

Création de ressources Amazon QLDB avecAWS CloudFormation

Amazon QLDB est intégré àAWS CloudFormation, un service qui vous aide à modéliser et à configurer vosAWS ressources afin que vous puissiez consacrer moins de temps à la création et à la gestion de vos ressources et de votre infrastructure. Vous créez un modèle qui décrit toutes lesAWS ressources que QLDB, etAWS CloudFormation alloue et configure ces ressources.

Lorsque vous utilisezAWS CloudFormation, vous pouvez réutiliser votre modèle pour configurer vos ressources QLDB de manière cohérente et répétée. Décrivez vos ressources une seule fois, puis allouez-les autant de fois que vous le souhaitez dans plusieurs Comptes AWS et régions.

QLDB etAWS CloudFormation modèles

Pour mettre en service et configurer des ressources pour QLDB et les services associés, vous devez maîtriser les AWS CloudFormationmodèles. Les modèles sont des fichiers texte formatés en JSON ou YAML. Ces modèles décrivent les ressources que vous souhaitez allouer dans vos piles AWS CloudFormation. Si JSON ou YAML ne vous est pas familier, vous pouvez utiliser AWS CloudFormation Designer pour vous aider à démarrer avec des modèles AWS CloudFormation. Pour plus d'informations, consultez Qu'est-ce que AWS CloudFormation Designer ? dans le AWS CloudFormationGuide de l'utilisateur.

QLDB prend en charge la création de livres et de flux de journaux dansAWS CloudFormation. Pour de plus amples informations, y compris des exemples de modèles JSON et YAML pour les registres et les flux, consultez les références de type de ressource suivantes dans le Guide deAWS CloudFormation l'utilisateur :

- AWS: :QLDB : :Ledger

- AWS: :QLDB : :Stream Référence

En savoir plus sur AWS CloudFormation

Pour en savoir plus sur AWS CloudFormation, consultez les ressources suivantes :

- AWS CloudFormation
- Guide de l'utilisateur AWS CloudFormation
- Référence API AWS CloudFormation
- Guide de l'utilisateur de l'interface de ligne de commande AWS CloudFormation

Balisage des ressources Amazon QLDB

Une balise est un attribut personnalisé que vous conférez ou que AWS attribue à une ressource AWS. Chaque balise se compose de deux parties :

- Une clé de balise (par exemple, `CostCenter`, `Environment` ou `Project`). Les clés de balises sont sensibles à la casse.
- Un champ facultatif appelé valeur de balise (par exemple, `111122223333` ou `Production`). Si la valeur de balise est identique à l'utilisation d'une chaîne vide. Les valeurs de balise sont sensibles à la casse, tout comme les clés de balise.

Les balises vous permettent d'effectuer les actions suivantes :

- Identifier et organiser vos ressources AWS. De nombreux services Services AWS prennent en charge l'indexation. Vous pouvez donc attribuer le même indexe à des ressources appartenant à différents services, afin d'indiquer que les ressources sont liées. Par exemple, vous pouvez affecter à un registre Amazon QLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL
- Suivre vos coûts AWS. Vous activez ces balises sur le tableau de bord AWS Billing and Cost Management. AWS utilise les balises pour classer vos coûts et pour vous fournir un rapport mensuel d'allocation des coûts. Pour plus d'informations, consultez Utilisation des étiquettes d'allocation des coûts dans le AWS Billing Guide de l'utilisateur.
- Contrôlez l'accès à vosAWS ressources avecAWS Identity and Access Management (IAM). Pour plus d'informations, consultezContrôle d'accès basé sur les attributs (ABAC) avec QLDB ce guide du développeur et Contrôlez l'accès à l'aide de balises IAM dans le Guide de l'utilisateur IAM.

Pour obtenir des conseils sur l'utilisation des balises, consultez l'article Stratégies de balisage AWS sur le blog AWS Answers.

Les sections suivantes fournissent de plus amples informations sur les balises pour Amazon QLDB

Rubriques

Ressources prises en charge dans Amazon QLDB

Les ressources suivantes dans Amazon QLDB

- grand livre
- table
- flux de journaux

Pour obtenir des informations sur l'ajout et la gestion de balises, veuillez consulter Gestion des balises.

Conventions de dénomination et d'utilisation des balises

Les conventions d'utilisation et d'attribution de noms de base suivantes s'appliquent à l'utilisation des balises avec les ressources Amazon QLDB

- Chaque ressource peut avoir un maximum de 50 balises.
- Pour chaque ressource, chaque clé de balise doit être unique, et chaque clé de balise peut avoir une seule valeur.
- La longueur maximale des clés de balise est de 128 caractères Unicode en UTF-8.
- La longueur maximale des valeurs de balise est de 256 caractères Unicode en UTF-8.
- Les caractères autorisés sont les lettres, les espaces et les chiffres représentables en UTF-8, ainsi que les caractères spéciaux suivants : . : + = @ _ / - (tiret).
- Les clés et valeurs de balise sont sensibles à la casse. La bonne pratique consiste à choisir une stratégie pour mettre des balises en majuscule et mettre en œuvre cette stratégie de manière

cohérente sur tous les types de ressources. Par exemple, décidez si vous souhaitez utiliser `Costcenter`, `costcenter` ou `CostCenter`, et utilisez la même convention pour toutes les balises. Évitez d'utiliser des balises avec une incohérence de traitement de cas similaires.

- Le préfixe `aws:` est réservé à l'utilisation d'AWS. Vous ne pouvez pas modifier ou supprimer la clé ou la valeur d'une balise lorsque la balise possède une clé de balise avec le préfixe `aws:`. Les étiquettes avec ce préfixe ne sont pas comptabilisées comme vos étiquettes pour la limite de ressources.

Gestion des balises

Les balises sont constituées des propriétés `Key` et `Value` sur une ressource. Vous pouvez utiliser la console Amazon QLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLAWS CLI Vous pouvez également utiliser l'éditeur deAWS Resource Groups balises pour gérer les balises.

Pour obtenir plus d'informations sur l'utilisation des balises, consultez les opérations d'API suivantes :

- ListTagsForResourcedans le manuel de référence de l'API Amazon QLDB
- TagResourcedans le manuel de référence de l'API Amazon QLDB
- UntagResourcedans le manuel de référence de l'API Amazon QLDB

Pour utiliser le panneau de balisage QLDB (console)

1. Connectez-vous auAWS Management Console et ouvrez la console Amazon QLDB https://console.aws.amazon.com/qldb
2. Dans le panneau de navigation, choisissez Registre Amazon LDB.
3. Dans la liste des livres, choisissez le nom du livre dont vous souhaitez gérer les balises.
4. Sur la page des détails du livre, recherchez la fiche Tags et choisissez Gérer les balises.
5. Sur la page Gérer les balises, vous pouvez ajouter, modifier ou supprimer toutes les balises adaptées à votre registre. Lorsque les clés et les valeurs vous conviennent, choisissez Enregistrer.

Balisage des ressources au moment de la création

Pour les ressources QLDB qui prennent en charge le balisage, vous pouvez définir des balises lors de la création de la ressource à l'AWS Management Consoleaide de l'APIAWS CLI, de ou de l'API

QLDB. En attribuant des étiquettes aux ressources au moment de la création, vous pouvez supprimer la nécessité d'exécuter des scripts d'identification personnalisés après la création de ressources.

Une fois qu'une ressource est balisée, vous pouvez contrôler l'accès à la ressource en fonction de ces balises. Par exemple, vous pouvez accorder un accès complet uniquement aux ressources de table qui possèdent une balise spécifique. Pour un exemple de politique JSON, reportez-vous à la sectionAccès complet à toutes les actions en fonction des balises du tableau.

> **Note**
>
> Les ressources de table et de flux n'héritent pas des balises de leur ressource de registre racine.

Vous pouvez également définir des balises de table en les spécifiant dans une instruction`CREATE TABLE` PartiQL. Pour en savoir plus, consultez la section Tableaux de balisage.

Sécurité dans Amazon QLDB

La sécurité du cloud AWS est la priorité absolue. En tant que AWS client, vous bénéficiez d'un centre de données et d'une architecture réseau conçus pour répondre aux exigences des entreprises les plus sensibles en matière de sécurité.

La sécurité est une responsabilité partagée entre vous AWS et vous. Le modèle de responsabilité partagée décrit cette notion par les termes sécurité du cloud et sécurité dans le cloud :

- Sécurité du cloud : AWS est chargée de protéger l'infrastructure qui s'exécute Services AWS dans le AWS Cloud. AWS vous fournit également des services que vous pouvez utiliser en toute sécurité. Des auditeurs tiers testent et vérifient régulièrement l'efficacité de notre sécurité dans le cadre des programmes de conformitéAWS. Pour en savoir plus sur les programmes de conformité qui s'appliquent à Amazon QLDB, AWS consultez la section Services concernés par programme de conformité.
- Sécurité dans le cloud — Votre responsabilité est déterminée par Service AWS ce que vous utilisez. Vous êtes également responsable d'autres facteurs, y compris la sensibilité de vos données, les exigences de votre entreprise, ainsi que la législation et la réglementation applicables.

Cette documentation vous aide à comprendre comment appliquer le modèle de responsabilité partagée lors de l'utilisation de QLDB. Les rubriques suivantes expliquent comment configurer QLDB pour répondre à vos objectifs de sécurité et de conformité. Vous apprendrez également à utiliser d'autres outils Services AWS qui vous aideront à surveiller et à sécuriser vos ressources QLDB.

Rubriques

Protection des données dans Amazon QLDB

Le modèle de responsabilité AWS partagée Le modèle s'applique à la protection des données dans Amazon QLDB. Comme décrit dans ce modèle, AWS est chargé de protéger l'infrastructure mondiale qui gère tous les AWS Cloud. La gestion du contrôle de votre contenu hébergé sur cette infrastructure relève de votre responsabilité. Vous êtes également responsable des tâches de configuration et de gestion de la sécurité des Services AWS que vous utilisez. Pour plus d'informations sur la confidentialité des données, consultez Questions fréquentes (FAQ) sur la confidentialité des données. Pour en savoir plus sur la protection des données en Europe, consultez le billet de blog Modèle de responsabilité partagée AWS et RGPD (Règlement général sur la protection des données) sur le Blog de sécuritéAWS .

À des fins de protection des données, nous vous recommandons de protéger les Compte AWS informations d'identification et de configurer les utilisateurs individuels avec AWS IAM Identity Center ou AWS Identity and Access Management (IAM). Ainsi, chaque utilisateur se voit attribuer uniquement les autorisations nécessaires pour exécuter ses tâches. Nous vous recommandons également de sécuriser vos données comme indiqué ci-dessous :

- Utilisez l'authentification multifactorielle (MFA) avec chaque compte.
- Utilisez le protocole SSL/TLS pour communiquer avec les ressources. AWS Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Configurez l'API et la journalisation de l'activité des utilisateurs avec AWS CloudTrail.
- Utilisez des solutions de AWS chiffrement, ainsi que tous les contrôles de sécurité par défaut qu'ils contiennent Services AWS.
- Utilisez des services de sécurité gérés avancés tels qu'Amazon Macie, qui contribuent à la découverte et à la sécurisation des données sensibles stockées dans Amazon S3.
- Si vous avez besoin de modules cryptographiques validés par la norme FIPS 140-2 pour accéder AWS via une interface de ligne de commande ou une API, utilisez un point de terminaison FIPS. Pour plus d'informations sur les points de terminaison FIPS (Federal Information Processing Standard) disponibles, consultez Federal Information Processing Standard (FIPS) 140-2 (Normes de traitement de l'information fédérale).

Nous vous recommandons fortement de ne jamais placer d'informations confidentielles ou sensibles, telles que les adresses e-mail de vos clients, dans des balises ou des champs de texte libre tels que le champ Name (Nom). Cela inclut lorsque vous travaillez avec QLDB ou un Services AWS autre outil à l'aide de la console, de l'API AWS CLI ou des SDK. AWS Toutes les données que vous entrez dans

des balises ou des champs de texte de forme libre utilisés pour les noms peuvent être utilisées à des fins de facturation ou dans les journaux de diagnostic. Si vous fournissez une adresse URL à un serveur externe, nous vous recommandons fortement de ne pas inclure d'informations d'identification dans l'adresse URL permettant de valider votre demande adressée à ce serveur.

> **Note**
>
> Ces conseils pour éviter les balises ou les champs de forme libre pour les informations sensibles font référence aux métadonnées d'une ressource de registre QLDB, plutôt qu'aux données stockées dans le registre. Vos données stockées dans une ressource de registre QLDB ne sont pas utilisées pour les journaux de facturation ou de diagnostic.

Rubriques

Chiffrement au repos dans Amazon QLDB

Toutes les données stockées dans Amazon QLDB sont entièrement chiffrées au repos par défaut. Le chiffrement QLDB au repos fournit une sécurité renforcée en chiffrant toutes les données du registre au repos à l'aide des clés de chiffrement dans (). AWS Key Management Service AWS KMS Cette fonctionnalité réduit la lourdeur opérationnelle et la complexité induites par la protection des données sensibles. Avec le chiffrement au repos, vous pouvez créer des applications de registre sensibles à la sécurité qui répondent aux exigences réglementaires et de conformité strictes en matière de chiffrement.

Le chiffrement au repos s'intègre AWS KMS à la gestion de la clé de chiffrement utilisée pour protéger vos registres QLDB. Pour plus d'informations AWS KMS, consultez AWS Key Management Service les concepts du Guide du AWS Key Management Service développeur.

Dans QLDB, vous pouvez spécifier le type de pour chaque ressource AWS KMS key de registre. Lorsque vous créez un nouveau registre ou que vous mettez à jour un registre existant, vous pouvez choisir l'un des types de clés KMS suivants pour protéger les données de votre registre :

- Clé détenue par AWS— Type de cryptage par défaut. La clé appartient à QLDB (sans frais supplémentaires).

- Clé gérée par le client : la clé est stockée dans votre ordinateur Compte AWS et vous l'avez créée, détenue et gérée. Vous avez le contrôle total de la clé (AWS KMS des frais s'appliquent).

> **Note**
>
> Amazon QLDB a lancé le support pour la AWS KMS keys gestion des clients le 22 juillet 2021. Tous les registres créés avant le lancement sont protégés Clés détenues par AWS par défaut, mais ne sont actuellement pas éligibles au chiffrement au repos à l'aide de clés gérées par le client.
> Vous pouvez consulter l'heure de création de votre registre sur la console QLDB.

Lorsque vous accédez à un registre, QLDB déchiffre les données de manière transparente. Vous pouvez basculer entre la clé gérée par le client Clé détenue par AWS et la clé gérée par le client à tout moment. Il n'est pas nécessaire de modifier le code ou les applications pour utiliser ou gérer des données chiffrées.

Vous pouvez spécifier une clé de chiffrement lorsque vous créez un nouveau registre ou que vous modifiez la clé de chiffrement d'un registre existant à l'aide de l' AWS Management Console API QLDB ou du (). AWS Command Line Interface AWS CLI Pour plus d'informations, consultez Utilisation de clés gérées par le client dans Amazon QLDB.

> **Note**
>
> Par défaut, Amazon QLDB active automatiquement le chiffrement au repos, sans frais Clés détenues par AWS supplémentaires. Toutefois, AWS KMS des frais s'appliquent pour l'utilisation d'une clé gérée par le client. Pour plus d'informations sur la tarification, consultez Tarification AWS Key Management Service.
> Le chiffrement QLDB au repos est disponible partout Régions AWS où QLDB est disponible.

Rubriques

- Chiffrement au repos : comment cela fonctionne dans Amazon QLDB
- Utilisation de clés gérées par le client dans Amazon QLDB

Chiffrement au repos : comment cela fonctionne dans Amazon QLDB

Le chiffrement QLDB au repos chiffre vos données à l'aide de la norme de chiffrement avancée 256 bits (AES-256). Cela permet de sécuriser vos données contre tout accès non autorisé au stockage sous-jacent. Toutes les données stockées dans les registres QLDB sont chiffrées au repos par défaut. Le chiffrement côté serveur est transparent, ce qui signifie qu'il n'est pas nécessaire de modifier les applications.

Encryption at rest s'intègre à AWS Key Management Service (AWS KMS) pour gérer la clé de chiffrement utilisée pour protéger vos registres QLDB. Lorsque vous créez un nouveau registre ou que vous mettez à jour un registre existant, vous pouvez choisir l'un des types de AWS KMS clés suivants :

- Clé détenue par AWS— Type de cryptage par défaut. La clé appartient à QLDB (sans frais supplémentaires).
- Clé gérée par le client : la clé est stockée dans votre ordinateur Compte AWS et vous l'avez créée, détenue et gérée. Vous avez le contrôle total de la clé (AWS KMS des frais s'appliquent).

Rubriques

Clé détenue par AWS

Clés détenues par AWS ne sont pas stockés dans votre Compte AWS. Elles font partie d'un ensemble de clés KMS qui AWS possède et gère pour une utilisation multiple Comptes AWS. Services AWS peut être utilisé Clés détenues par AWS pour protéger vos données.

Vous n'avez pas besoin de créer ou de gérer Clés détenues par AWS. Toutefois, vous ne pouvez ni consulter, ni suivre Clés détenues par AWS, ni auditer leur utilisation. Aucuns frais mensuels ni frais d'utilisation ne vous sont facturés Clés détenues par AWS, et ils ne sont pas pris en compte dans les AWS KMS quotas de votre compte.

Pour plus d'informations, consultez Clés détenues par AWS dans le Guide du développeur AWS Key Management Service .

Clé gérée par le client

Les clés gérées par le client sont des clés KMS Compte AWS que vous créez, détenez et gérez. Vous avez un contrôle total sur ces clés KMS. QLDB prend uniquement en charge les clés KMS de chiffrement symétriques.

Utilisez une clé gérée par le client pour obtenir les fonctions suivantes.

- Définition et gestion des politiques clés, des politiques IAM et des autorisations pour contrôler l'accès à la clé
- Activation et désactivation de la clé
- Matériau cryptographique rotatif pour la clé
- Création de tags clés et d'alias
- Planification de la suppression de la clé
- Importer votre propre matériel clé ou utiliser un magasin de clés personnalisé que vous possédez et gérez
- Utilisation AWS CloudTrail d'Amazon CloudWatch Logs pour suivre les demandes auxquelles QLDB envoie AWS KMS en votre nom

Pour plus d'informations, consultez Clés gérées par le client dans le Guide du développeur AWS Key Management Service (langue française non garantie).

Les clés gérées par le client sont facturées pour chaque appel d'API, et AWS KMS des quotas s'appliquent à ces clés KMS. Pour plus d'informations, consultez la section Quotas de AWS KMS ressources ou de demandes.

Lorsque vous spécifiez une clé gérée par le client comme clé KMS pour un registre, toutes les données du registre stockées dans les journaux et dans le stockage indexé sont protégées par la même clé gérée par le client.

Clés gérées par le client inaccessibles

Si vous désactivez votre clé gérée par le client, planifiez la suppression de la clé ou révoquez les autorisations relatives à la clé, le statut du chiffrement de votre registre devient le même. `KMS_KEY_INACCESSIBLE` Dans cet état, le registre est altéré et n'accepte aucune demande de

lecture ou d'écriture. Une clé inaccessible empêche tous les utilisateurs et le service QLDB de chiffrer ou de déchiffrer les données et d'effectuer des opérations de lecture et d'écriture dans le registre. QLDB doit avoir accès à votre clé KMS pour que vous puissiez continuer à accéder à votre registre et pour éviter toute perte de données.

> **Important**
>
> Un registre altéré revient automatiquement à l'état actif une fois que vous avez rétabli les autorisations sur la clé ou après avoir réactivé la clé qui a été désactivée.
> Cependant, la suppression d'une clé gérée par le client est irréversible. Une fois qu'une clé est supprimée, vous ne pouvez plus accéder aux registres protégés par cette clé, et les données deviennent définitivement irrécupérables.

Pour vérifier l'état de chiffrement d'un registre, utilisez l'opération AWS Management Console ou l'DescribeLedgerAPI.

Comment Amazon QLDB utilise les subventions dans AWS KMS

QLDB nécessite des autorisations pour utiliser votre clé gérée par le client. Lorsque vous créez un registre protégé par une clé gérée par le client, QLDB crée des subventions en votre nom en envoyant des demandes à. CreateGrant AWS KMS AWS KMS Les autorisations sont utilisées pour donner à QLDB l'accès à une clé KMS chez un client. Compte AWS Pour plus d'informations, consultez Utilisation des attributions dans le Guide du développeur AWS Key Management Service .

QLDB a besoin des autorisations nécessaires pour utiliser votre clé gérée par le client pour les opérations suivantes : AWS KMS

- DescribeKey— Vérifiez que la clé KMS de chiffrement symétrique spécifiée est valide.
- GenerateDataKey— Générez une clé de données symétrique unique que QLDB utilise pour chiffrer les données au repos dans votre registre.
- Déchiffrer : déchiffrez la clé de données chiffrée par votre clé gérée par le client.
- Chiffrer — Chiffrez le texte brut en texte chiffré à l'aide de la clé gérée par le client.

Vous pouvez révoquer une autorisation pour supprimer l'accès du service à la clé gérée par le client à tout moment. Dans ce cas, la clé devient inaccessible et QLDB perd l'accès à toutes les données du registre protégées par la clé gérée par le client. Dans cet état, le registre est altéré et n'accepte aucune demande de lecture ou d'écriture tant que vous n'avez pas rétabli les autorisations sur la clé.

Rétablir les subventions dans AWS KMS

Pour rétablir les autorisations sur une clé gérée par le client et récupérer l'accès à un registre dans QLDB, vous pouvez mettre à jour le registre et spécifier la même clé KMS. Pour obtenir des instructions, veuillez consulter Mise à jour AWS KMS key d'un registre existant.

Considérations relatives au chiffrement au repos

Tenez compte des points suivants lorsque vous utilisez le chiffrement au repos dans QLDB :

- Le chiffrement au repos côté serveur est activé par défaut sur toutes les données du registre QLDB et ne peut pas être désactivé. Vous ne pouvez pas chiffrer uniquement un sous-ensemble de données dans un registre.
- La fonction de chiffrement au repos chiffre uniquement les données lorsque qu'elles sont statiques (au repos) sur un support de stockage permanent. Si la sécurité des données est un problème pour les données en transit ou en cours d'utilisation, vous devrez peut-être prendre les mesures supplémentaires suivantes :
  - Données en transit : toutes vos données dans QLDB sont cryptées pendant le transfert. Par défaut, les communications à destination et en provenance de QLDB utilisent le protocole HTTPS, qui protège le trafic réseau en utilisant le cryptage SSL (Secure Sockets Layer) / Transport Layer Security (TLS).
  - Données en cours d'utilisation : protégez vos données avant de les envoyer à QLDB en utilisant le chiffrement côté client.

Pour savoir comment implémenter des clés gérées par le client pour les registres, passez àUtilisation de clés gérées par le client dans Amazon QLDB.

Utilisation de clés gérées par le client dans Amazon QLDB

Vous pouvez utiliser l'API AWS Management Console, le AWS Command Line Interface (AWS CLI) ou l'API QLDB pour spécifier les nouveaux registres et AWS KMS key les registres existants dans Amazon QLDB. Les rubriques suivantes décrivent comment gérer et surveiller l'utilisation de vos clés gérées par le client dans QLDB.

Rubriques

- Surveillance de votre AWS KMS keys

Prérequis

Avant de pouvoir protéger un registre QLDB avec une clé gérée par le client, vous devez d'abord créer la clé dans (). AWS Key Management Service AWS KMS Vous devez également spécifier une politique clé qui permet à QLDB de créer des subventions à AWS KMS key ce sujet en votre nom.

Création d'une clé gérée par le client

Pour créer une clé gérée par le client, suivez les étapes décrites dans la section Création de clés KMS de chiffrement symétriques dans le guide du AWS Key Management Service développeur. QLDB ne prend pas en charge les clés asymétriques.

Définition d'une politique de clé

Les politiques clés constituent le principal moyen de contrôler l'accès aux clés gérées par le client dans AWS KMS. Chaque clé gérée par le client doit avoir exactement une politique clé. Les déclarations dans le document de politique de clé déterminent qui a l'autorisation d'utiliser la clé KMS et la façon dont ils peuvent l'utiliser. Pour plus d'informations, consultez la section Utilisation de politiques clés dans AWS KMS.

Vous pouvez définir une politique clé lorsque vous créez votre clé gérée par le client. Pour modifier une politique clé pour une clé gérée par le client existante, voir Modifier une politique clé.

Pour autoriser QLDB à utiliser votre clé gérée par le client, la politique en matière de clés doit inclure des autorisations pour les actions suivantes : AWS KMS

- kms : CreateGrant — Ajoute une autorisation à une clé gérée par le client. Accorde un accès de contrôle à une clé KMS spécifiée.

  Lorsque vous créez ou mettez à jour un registre avec une clé gérée par le client spécifiée, QLDB crée des autorisations qui permettent d'accéder aux opérations de subvention dont elle a besoin. Les opérations de subvention incluent les suivantes :

  - GenerateDataKey
  - Decrypt (Déchiffrer)
  - Encrypt (Chiffrer)
- kms : DescribeKey — Renvoie des informations détaillées sur une clé gérée par le client. QLDB utilise ces informations pour valider la clé.

Exemple de politique clé

Voici un exemple de politique clé que vous pouvez utiliser pour QLDB. Cette politique permet aux principaux autorisés à utiliser QLDB depuis le `111122223333` compte d'appeler les opérations `CreateGrant` et sur `DescribeKey` la ressource. `arn:aws:kms:us-east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`

Pour utiliser cette politique, remplacez us-east-1, 111122223333 et 1234abcd-12ab-34cd-56ef-1234567890ab dans l'exemple par vos propres informations.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid" : "Allow access to principals authorized to use Amazon QLDB",
            "Effect" : "Allow",
            "Principal" : {
                "AWS" : "*"
            },
            "Action" : [
                "kms:DescribeKey",
                "kms:CreateGrant"
            ],
            "Resource" : "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "Condition" : {
                "StringEquals" : {
                    "kms:ViaService" : "qldb.us-east-1.amazonaws.com",
                    "kms:CallerAccount" : "111122223333"
                }
            }
        }
    ]
}
```

Spécifier le AWS KMS key pour un nouveau registre

Procédez comme suit pour spécifier une clé KMS lorsque vous créez un nouveau registre à l'aide de la console QLDB ou du. AWS CLI

Vous pouvez spécifier une clé gérée par le client à l'aide d'un ID, d'un alias ou d'un Amazon Resource Name (ARN). Pour en savoir plus, consultez la section Identifiants clés (KeyId) dans le guide du AWS Key Management Service développeur.

> **Note**
>
> Les clés inter-régions ne sont pas prises en charge. La clé KMS spécifiée doit être identique Région AWS à celle de votre registre.

Création d'un registre (console)

1. Connectez-vous à la AWS Management Console console Amazon QLDB et ouvrez-la à l'adresse https://console.aws.amazon.com/qldb.
2. Choisissez Create Ledger.
3. Sur la page Créer un registre, procédez comme suit :
   - Informations sur le registre — Entrez un nom de registre unique parmi tous les livres du livre actuel Compte AWS et de la région.
   - Mode autorisations — Choisissez un mode d'autorisations à attribuer au registre :
     - Tout autoriser
     - Standard (recommandé)
   - Chiffrer les données au repos : choisissez le type de clé KMS à utiliser pour le chiffrement au repos :
     - Utiliser une clé KMS AWS possédée — Utilisez une clé KMS détenue et gérée par vous AWS en votre nom. Il s'agit de l'option par défaut qui ne nécessite aucune configuration supplémentaire.
     - Choisissez une autre AWS KMS clé : utilisez une clé KMS de chiffrement symétrique dans le compte que vous créez, possédez et gérez.

       Pour créer une nouvelle clé à l'aide de la AWS KMS console, choisissez Créer une AWS KMS clé. Pour plus d'informations, consultez Création de clés KMS de chiffrement symétriques dans le Guide du développeur AWS Key Management Service .

       Pour utiliser une clé KMS existante, choisissez-en une dans la liste déroulante ou spécifiez un ARN de clé KMS.
4. Lorsque les paramètres sont tels que vous le souhaitez, choisissez Create ledger.

Vous pouvez accéder à votre registre QLDB lorsque son statut devient Actif. Cela peut prendre plusieurs minutes.

Création d'un registre ()AWS CLI

Utilisez le AWS CLI pour créer un registre dans QLDB avec la clé Clé détenue par AWS par défaut ou une clé gérée par le client.

Example — Pour créer un registre avec la valeur par défaut Clé détenue par AWS

```
aws qldb create-ledger --name my-example-ledger --permissions-mode STANDARD
```

Example — Pour créer un registre avec une clé gérée par le client

```
aws qldb create-ledger \
    --name my-example-ledger \
    --permissions-mode STANDARD \
    --kms-key arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

Mise à jour AWS KMS key d'un registre existant

Vous pouvez également utiliser la console QLDB ou mettre à jour AWS CLI la clé KMS d'un registre existant en une clé gérée par le client à Clé détenue par AWS tout moment.

> **Note**
>
> Amazon QLDB a lancé le support pour la AWS KMS keys gestion des clients le 22 juillet 2021. Tous les registres créés avant le lancement sont protégés Clés détenues par AWS par défaut, mais ne sont actuellement pas éligibles au chiffrement au repos à l'aide de clés gérées par le client.
> Vous pouvez consulter l'heure de création de votre registre sur la console QLDB.

Les principales modifications apportées à QLDB sont asynchrones. Le registre est entièrement accessible sans aucun impact sur les performances pendant le traitement de la modification clé. Le temps nécessaire pour mettre à jour une clé varie en fonction de la taille du registre.

Vous pouvez spécifier une clé gérée par le client à l'aide d'un ID, d'un alias ou d'un Amazon Resource Name (ARN). Pour en savoir plus, consultez la section Identifiants clés (KeyId) dans le guide du AWS Key Management Service développeur.

> **Note**
>
> Les clés inter-régions ne sont pas prises en charge. La clé KMS spécifiée doit être identique Région AWS à celle de votre registre.

Mettre à jour un registre (console)

1. Connectez-vous à la AWS Management Console console Amazon QLDB et ouvrez-la à l'adresse https://console.aws.amazon.com/qldb.
2. Dans le volet de navigation, choisissez Ledgers.
3. Dans la liste des livres, sélectionnez le livre que vous souhaitez mettre à jour, puis choisissez Modifier le registre.
4. Sur la page Modifier le registre, choisissez le type de clé KMS à utiliser pour le chiffrement au repos :

   - Utiliser une clé KMS AWS possédée — Utilisez une clé KMS détenue et gérée par vous AWS en votre nom. Il s'agit de l'option par défaut qui ne nécessite aucune configuration supplémentaire.
   - Choisissez une autre AWS KMS clé : utilisez une clé KMS de chiffrement symétrique dans le compte que vous créez, possédez et gérez.

     Pour créer une nouvelle clé à l'aide de la AWS KMS console, choisissez Créer une AWS KMS clé. Pour plus d'informations, consultez Création de clés KMS de chiffrement symétriques dans le Guide du développeur AWS Key Management Service .

     Pour utiliser une clé KMS existante, choisissez-en une dans la liste déroulante ou spécifiez un ARN de clé KMS.
5. Choisissez Confirmer les modifications.

Mettre à jour un registre ()AWS CLI

Utilisez le AWS CLI pour mettre à jour un registre existant dans QLDB avec la clé Clé détenue par AWS par défaut ou une clé gérée par le client.

Example — Pour mettre à jour un registre avec la valeur par défaut Clé détenue par AWS

```
aws qldb update-ledger --name my-example-ledger --kms-key AWS_OWNED_KMS_KEY
```

Example — Pour mettre à jour un registre avec une clé gérée par le client

```
aws qldb update-ledger \
    --name my-example-ledger \
    --kms-key arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

Surveillance de votre AWS KMS keys

Si vous utilisez une clé gérée par le client pour protéger vos registres Amazon QLDB, vous pouvez utiliser CloudWatch Amazon Logs pour AWS CloudTrailsuivre les demandes auxquelles QLDB envoie en votre nom. AWS KMS Pour plus d'informations, consultez la section Surveillance AWS KMS keys dans le guide du AWS Key Management Service développeur.

Les exemples suivants sont des entrées de CloudTrail journal pour les opérations `CreateGrant`,`GenerateDataKey`,`Decrypt`,`Encrypt`, et`DescribeKey`.

CreateGrant

Lorsque vous spécifiez une clé gérée par le client pour protéger votre registre, QLDB `CreateGrant` envoie des demandes en votre nom pour autoriser l'accès AWS KMS à votre clé KMS. En outre, QLDB utilise cette opération pour supprimer `RetireGrant` des autorisations lorsque vous supprimez un registre.

Les autorisations créées par QLDB sont spécifiques à un registre. Le principal de la `CreateGrant` demande est l'utilisateur qui a créé la table.

L'événement qui enregistre l'opération `CreateGrant` est similaire à l'exemple d'événement suivant. Les paramètres incluent l'Amazon Resource Name (ARN) de la clé gérée par le client, le principal du bénéficiaire et le principal sortant (le service QLDB), ainsi que les opérations couvertes par la subvention.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AKIAIOSFODNN7EXAMPLE:sample-user",
```

```
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/sample-user",
        "accountId": "111122223333",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AKIAIOSFODNN7EXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-06-04T21:37:11Z"
            }
        },
        "invokedBy": "qldb.amazonaws.com"
    },
    "eventTime": "2021-06-04T21:40:00Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "qldb.amazonaws.com",
    "userAgent": "qldb.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "granteePrincipal": "qldb.us-west-2.amazonaws.com",
        "operations": [
            "DescribeKey",
            "GenerateDataKey",
            "Decrypt",
            "Encrypt"
        ],
        "retiringPrincipal": "qldb.us-west-2.amazonaws.com"
    },
    "responseElements": {
        "grantId":
 "b3c83f999187ccc0979ef2ff86a1572237b6bba309c0ebce098c34761f86038a"
    },
    "requestID": "e99188d7-3b82-424e-b63e-e086d848ed60",
    "eventID": "88dc7ba5-4952-4d36-9ca8-9ab5d9598bab",
    "readOnly": false,
```

```
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333"
}
```

GenerateDataKey

Lorsque vous spécifiez une clé gérée par le client pour protéger votre registre, QLDB crée une clé de données unique. Il envoie une `GenerateDataKey` demande AWS KMS qui spécifie la clé gérée par le client pour le registre.

L'événement qui enregistre l'opération `GenerateDataKey` est similaire à l'exemple d'événement suivant. L'utilisateur est le compte de service QLDB. Les paramètres incluent l'ARN de la clé gérée par le client, un spécificateur de clé de données qui nécessite une longueur de 32 octets et le contexte de chiffrement qui identifie le nœud interne de hiérarchie des clés.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "qldb.amazonaws.com"
    },
    "eventTime": "2021-06-04T21:40:01Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "qldb.amazonaws.com",
    "userAgent": "qldb.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "numberOfBytes": 32,
        "encryptionContext": {
            "key-hierarchy-node-id": "LY4HWMnkeZWKYi6MlitVJC",
```

```
            "key-hierarchy-node-version": "1"
        }
    },
    "responseElements": null,
    "requestID": "786977c9-e77c-467a-bff5-9ad5124a4462",
    "eventID": "b3f082cb-3e75-454e-bf0a-64be13075436",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333",
    "sharedEventID": "26688de5-0b1c-43d3-bc4f-a18029b08446"
}
```

Decrypt

Lorsque vous accédez à un registre, QLDB appelle `Decrypt` l'opération pour déchiffrer la clé de données stockée dans le registre afin de pouvoir accéder aux données cryptées du registre.

L'événement qui enregistre l'opération `Decrypt` est similaire à l'exemple d'événement suivant. L'utilisateur est le compte de service QLDB. Les paramètres incluent l'ARN de la clé gérée par le client et le contexte de chiffrement qui identifie le nœud interne de hiérarchie des clés.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "qldb.amazonaws.com"
    },
    "eventTime": "2021-06-04T21:40:56Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "qldb.amazonaws.com",
    "userAgent": "qldb.amazonaws.com",
```

```
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "encryptionContext": {
            "key-hierarchy-node-id": "LY4HWMnkeZWKYi6MlitVJC",
            "key-hierarchy-node-version": "1"
        }
    },
    "responseElements": null,
    "requestID": "28f2dd18-3cc1-4fe2-82f7-5154f4933ebf",
    "eventID": "603ad5d4-4744-4505-9c21-bd4a6cbd4b20",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333",
    "sharedEventID": "7b6ce3e3-a764-42ec-8f90-5418c97ec411"
}
```

Encrypt

QLDB appelle l'opération pour crypter `Encrypt` le texte brut en texte chiffré à l'aide de votre clé gérée par le client.

L'événement qui enregistre l'opération `Encrypt` est similaire à l'exemple d'événement suivant. L'utilisateur est le compte de service QLDB. Les paramètres incluent l'ARN de la clé gérée par le client et le contexte de chiffrement qui spécifie l'identifiant unique interne du registre.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "qldb.amazonaws.com"
    },
```

```
    "eventTime": "2021-06-04T21:40:01Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Encrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "qldb.amazonaws.com",
    "userAgent": "qldb.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "encryptionContext": {
            "LedgerId": "F6qRNziJLUXA4Vy2ZUv8YY"
        },
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT"
    },
    "responseElements": null,
    "requestID": "b2daca7d-4606-4302-a2d7-5b3c8d30c64d",
    "eventID": "b8aace05-2e37-4fed-ae6f-a45a1c6098df",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333",
    "sharedEventID": "ce420ab0-288e-4b4f-aee8-541e18a28aa5"
}
```

DescribeKey

QLDB appelle `DescribeKey` l'opération pour déterminer si la clé KMS que vous avez spécifiée existe dans Compte AWS la région et.

L'événement qui enregistre l'opération `DescribeKey` est similaire à l'exemple d'événement suivant. Le principal est l'utilisateur Compte AWS qui a spécifié la clé KMS. Les paramètres incluent l'ARN de la clé gérée par le client.

```
{
```

```
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AKIAIOSFODNN7EXAMPLE:sample-user",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/sample-user",
        "accountId": "111122223333",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AKIAIOSFODNN7EXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-06-04T21:37:11Z"
            }
        },
        "invokedBy": "qldb.amazonaws.com"
    },
    "eventTime": "2021-06-04T21:40:00Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DescribeKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "qldb.amazonaws.com",
    "userAgent": "qldb.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": null,
    "requestID": "a30586af-c783-4d25-8fda-33152c816c36",
    "eventID": "7a9caf07-2b27-44ab-afe4-b259533ebb88",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
```

```
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333"
}
```

Chiffrement en transit dans Amazon QLDB

Amazon QLDB accepte uniquement les connexions sécurisées utilisant le protocole HTTPS, qui protège le trafic réseau en utilisant le protocole SSL (Secure Sockets Layer) /Transport Layer Security (TLS). Le chiffrement en transit fournit une couche supplémentaire de protection des données en chiffrant vos données à destination et en provenance de QLDB. Les politiques organisationnelles, les réglementations sectorielles ou gouvernementales et les exigences de conformité nécessitent souvent l'utilisation du chiffrement en transit pour renforcer la sécurité des données de vos applications lorsqu'elles transmettent des données sur le réseau.

QLDB propose également des points de terminaison FIPS dans certaines régions. Contrairement aux points de AWS terminaison standard, les points de terminaison FIPS utilisent une bibliothèque logicielle TLS conforme à la norme fédérale de traitement de l'information (FIPS) 140-2. Ces points de terminaison peuvent être requis par les entreprises qui interagissent avec le gouvernement des États-Unis. Pour plus d'informations, consultez la section Points de terminaison FIPS dans le. Références générales AWS Pour obtenir la liste complète des régions et des points de terminaison disponibles pour QLDB, consultez la section Points de terminaison et quotas Amazon QLDB.

Identity and Access Management pour Amazon QLDB

AWS Identity and Access Management (IAM) est un outil Service AWS qui permet à un administrateur de contrôler en toute sécurité l'accès aux AWS ressources. Les administrateurs IAM contrôlent qui peut être authentifié (connecté) et autorisé (autorisé) à utiliser les ressources QLDB. IAM est un Service AWS outil que vous pouvez utiliser sans frais supplémentaires.

Rubriques

- Gestion des accès à l’aide de politiques
- Comment Amazon QLDB fonctionne avec IAM
- Commencer à utiliser le mode d'autorisation standard dans Amazon QLDB
- Exemples de politiques basées sur l'identité pour Amazon QLDB
- Prévention du cas de figure de l’adjoint désorienté entre services
- AWS politiques gérées pour Amazon QLDB
- Résolution des problèmes d'identité et d'accès à Amazon QLDB

Public ciblé

La façon dont vous utilisez AWS Identity and Access Management (IAM) varie en fonction du travail que vous effectuez dans QLDB.

Utilisateur du service : si vous utilisez le service QLDB pour effectuer votre travail, votre administrateur vous fournit les informations d'identification et les autorisations dont vous avez besoin. Au fur et à mesure que vous utilisez davantage de fonctionnalités QLDB pour effectuer votre travail, vous aurez peut-être besoin d'autorisations supplémentaires. En comprenant bien la gestion des accès, vous saurez demander les autorisations appropriées à votre administrateur. Si vous ne pouvez pas accéder à une fonctionnalité dans QLDB, consultez. Résolution des problèmes d'identité et d'accès à Amazon QLDB

Administrateur de service — Si vous êtes responsable des ressources QLDB dans votre entreprise, vous avez probablement un accès complet à QLDB. C'est à vous de déterminer les fonctionnalités et les ressources QLDB auxquelles les utilisateurs de votre service doivent accéder. Vous devez ensuite soumettre les demandes à votre administrateur IAM pour modifier les autorisations des utilisateurs de votre service. Consultez les informations sur cette page pour comprendre les concepts de base d’IAM. Pour en savoir plus sur la manière dont votre entreprise peut utiliser IAM avec QLDB, consultez. Comment Amazon QLDB fonctionne avec IAM

Administrateur IAM : si vous êtes administrateur IAM, vous souhaiterez peut-être en savoir plus sur la manière dont vous pouvez rédiger des politiques pour gérer l'accès à QLDB. Pour consulter des exemples de politiques basées sur l'identité QLDB que vous pouvez utiliser dans IAM, consultez. Exemples de politiques basées sur l'identité pour Amazon QLDB

Authentification par des identités

L'authentification est la façon dont vous vous connectez à AWS l'aide de vos informations d'identification. Vous devez être authentifié (connecté à AWS) en tant qu'utilisateur IAM ou en assumant un rôle IAM. Utilisateur racine d'un compte AWS

Vous pouvez vous connecter en AWS tant qu'identité fédérée en utilisant les informations d'identification fournies par le biais d'une source d'identité. AWS IAM Identity Center Les utilisateurs (IAM Identity Center), l'authentification unique de votre entreprise et vos informations d'identification Google ou Facebook sont des exemples d'identités fédérées. Lorsque vous vous connectez avec une identité fédérée, votre administrateur aura précédemment configuré une fédération d'identités avec des rôles IAM. Lorsque vous accédez à AWS l'aide de la fédération, vous assumez indirectement un rôle.

Selon le type d'utilisateur que vous êtes, vous pouvez vous connecter au portail AWS Management Console ou au portail AWS d'accès. Pour plus d'informations sur la connexion à AWS, consultez Comment vous connecter à votre compte Compte AWS dans le guide de Connexion à AWS l'utilisateur.

Si vous y accédez AWS par programmation, AWS fournit un kit de développement logiciel (SDK) et une interface de ligne de commande (CLI) pour signer cryptographiquement vos demandes à l'aide de vos informations d'identification. Si vous n'utilisez pas d' AWS outils, vous devez signer vous-même les demandes. Pour plus d'informations sur l'utilisation de la méthode recommandée pour signer vous-même les demandes, consultez la section Signature des demandes AWS d'API dans le guide de l'utilisateur IAM.

Quelle que soit la méthode d'authentification que vous utilisez, vous devrez peut-être fournir des informations de sécurité supplémentaires. Par exemple, il vous AWS recommande d'utiliser l'authentification multifactorielle (MFA) pour renforcer la sécurité de votre compte. Pour en savoir plus, consultez Authentification multifactorielle dans le Guide de l'utilisateur AWS IAM Identity Center et Utilisation de l'authentification multifactorielle (MFA) dans l'interface AWS dans le Guide de l'utilisateur IAM.

Compte AWS utilisateur root

Lorsque vous créez un Compte AWS, vous commencez par une identité de connexion unique qui donne un accès complet à toutes Services AWS les ressources du compte. Cette identité est appelée utilisateur Compte AWS root et est accessible en vous connectant avec l'adresse e-mail et le mot de passe que vous avez utilisés pour créer le compte. Il est vivement recommandé de ne pas

utiliser l'utilisateur racine pour vos tâches quotidiennes. Protégez vos informations d'identification d'utilisateur racine et utilisez-les pour effectuer les tâches que seul l'utilisateur racine peut effectuer. Pour obtenir la liste complète des tâches qui vous imposent de vous connecter en tant qu'utilisateur root, consultez Tâches nécessitant des informations d'identification d'utilisateur root dans le Guide de l'utilisateur IAM.

Identité fédérée

La meilleure pratique consiste à obliger les utilisateurs humains, y compris ceux qui ont besoin d'un accès administrateur, à utiliser la fédération avec un fournisseur d'identité pour accéder à l'aide Services AWS d'informations d'identification temporaires.

Une identité fédérée est un utilisateur de l'annuaire des utilisateurs de votre entreprise, d'un fournisseur d'identité Web AWS Directory Service, du répertoire Identity Center ou de tout utilisateur qui y accède à l'aide des informations d'identification fournies Services AWS par le biais d'une source d'identité. Lorsque des identités fédérées y accèdent Comptes AWS, elles assument des rôles, qui fournissent des informations d'identification temporaires.

Pour une gestion des accès centralisée, nous vous recommandons d'utiliser AWS IAM Identity Center. Vous pouvez créer des utilisateurs et des groupes dans IAM Identity Center, ou vous pouvez vous connecter et synchroniser avec un ensemble d'utilisateurs et de groupes dans votre propre source d'identité afin de les utiliser dans toutes vos applications Comptes AWS et applications. Pour obtenir des informations sur IAM Identity Center, consultez Qu'est-ce que IAM Identity Center ? dans le Guide de l'utilisateur AWS IAM Identity Center .

Utilisateurs et groupes IAM

Un utilisateur IAM est une identité au sein de vous Compte AWS qui possède des autorisations spécifiques pour une seule personne ou une seule application. Dans la mesure du possible, nous vous recommandons de vous appuyer sur des informations d'identification temporaires plutôt que de créer des utilisateurs IAM ayant des informations d'identification à long terme tels que les clés d'accès. Toutefois, si certains cas d'utilisation spécifiques nécessitent des informations d'identification à long terme avec les utilisateurs IAM, nous vous recommandons de faire pivoter les clés d'accès. Pour plus d'informations, consultez Rotation régulière des clés d'accès pour les cas d'utilisation nécessitant des informations d'identification dans le Guide de l'utilisateur IAM.

Un groupe IAM est une identité qui concerne un ensemble d'utilisateurs IAM. Vous ne pouvez pas vous connecter en tant que groupe. Vous pouvez utiliser les groupes pour spécifier des autorisations pour plusieurs utilisateurs à la fois. Les groupes permettent de gérer plus facilement les autorisations

pour de grands ensembles d'utilisateurs. Par exemple, vous pouvez avoir un groupe nommé IAMAdmins et accorder à ce groupe les autorisations d'administrer des ressources IAM.

Les utilisateurs sont différents des rôles. Un utilisateur est associé de manière unique à une personne ou une application, alors qu'un rôle est conçu pour être endossé par tout utilisateur qui en a besoin. Les utilisateurs disposent d'informations d'identification permanentes, mais les rôles fournissent des informations d'identification temporaires. Pour en savoir plus, consultez Quand créer un utilisateur IAM (au lieu d'un rôle) dans le Guide de l'utilisateur IAM.

Rôles IAM

Un rôle IAM est une identité au sein de vous Compte AWS dotée d'autorisations spécifiques. Le concept ressemble à celui d'utilisateur IAM, mais le rôle IAM n'est pas associé à une personne en particulier. Vous pouvez assumer temporairement un rôle IAM dans le en AWS Management Console changeant de rôle. Vous pouvez assumer un rôle en appelant une opération d' AWS API AWS CLI ou en utilisant une URL personnalisée. Pour plus d'informations sur les méthodes d'utilisation des rôles, consultez Utilisation de rôles IAM dans le Guide de l'utilisateur IAM.

Les rôles IAM avec des informations d'identification temporaires sont utiles dans les cas suivants :

- Accès utilisateur fédéré – Pour attribuer des autorisations à une identité fédérée, vous créez un rôle et définissez des autorisations pour le rôle. Quand une identité externe s'authentifie, l'identité est associée au rôle et reçoit les autorisations qui sont définies par celui-ci. Pour obtenir des informations sur les rôles pour la fédération, consultez Création d'un rôle pour un fournisseur d'identité tiers (fédération) dans le Guide de l'utilisateur IAM. Si vous utilisez IAM Identity Center, vous configurez un jeu d'autorisations. IAM Identity Center met en corrélation le jeu d'autorisations avec un rôle dans IAM afin de contrôler à quoi vos identités peuvent accéder après leur authentification. Pour plus d'informations sur les jeux d'autorisations, consultez la rubrique Jeux d'autorisations dans le Guide de l'utilisateur AWS IAM Identity Center .
- Autorisations d'utilisateur IAM temporaires : un rôle ou un utilisateur IAM peut endosser un rôle IAM pour profiter temporairement d'autorisations différentes pour une tâche spécifique.
- Accès intercompte : vous pouvez utiliser un rôle IAM pour permettre à un utilisateur (principal de confiance) d'un compte différent d'accéder aux ressources de votre compte. Les rôles constituent le principal moyen d'accorder l'accès intercompte. Toutefois, dans certains Services AWS cas, vous pouvez associer une politique directement à une ressource (au lieu d'utiliser un rôle comme proxy). Pour en savoir plus sur la différence entre les rôles et les politiques basées sur les ressources pour l'accès intercompte, consultez Différence entre les rôles IAM et les politiques basées sur les ressources dans le Guide de l'utilisateur IAM.

- Accès multiservices — Certains Services AWS utilisent des fonctionnalités dans d'autres Services AWS. Par exemple, lorsque vous effectuez un appel dans un service, il est courant que ce service exécute des applications dans Amazon EC2 ou stocke des objets dans Amazon S3. Un service peut le faire en utilisant les autorisations d'appel du principal, un rôle de service ou un rôle lié au service.
  - Sessions d'accès direct (FAS) : lorsque vous utilisez un utilisateur ou un rôle IAM pour effectuer des actions AWS, vous êtes considéré comme un mandant. Lorsque vous utilisez certains services, vous pouvez effectuer une action qui initie une autre action dans un autre service. FAS utilise les autorisations du principal appelant et Service AWS, associées Service AWS à la demande, pour adresser des demandes aux services en aval. Les demandes FAS ne sont effectuées que lorsqu'un service reçoit une demande qui nécessite des interactions avec d'autres personnes Services AWS ou des ressources pour être traitée. Dans ce cas, vous devez disposer d'autorisations nécessaires pour effectuer les deux actions. Pour plus de détails sur la politique relative à la transmission de demandes FAS, consultez Sessions de transmission d'accès.
  - Rôle de service : il s'agit d'un rôle IAM attribué à un service afin de réaliser des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer une fonction du service à partir d'IAM. Pour plus d'informations, consultez Création d'un rôle pour la délégation d'autorisations à un Service AWS dans le Guide de l'utilisateur IAM.
  - Rôle lié à un service — Un rôle lié à un service est un type de rôle de service lié à un. Service AWS Le service peut endosser le rôle afin d'effectuer une action en votre nom. Les rôles liés à un service apparaissent dans votre Compte AWS fichier et appartiennent au service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.
- Applications exécutées sur Amazon EC2 : vous pouvez utiliser un rôle IAM pour gérer les informations d'identification temporaires pour les applications qui s'exécutent sur une instance EC2 et qui envoient des demandes d'API. AWS CLI AWS Cette solution est préférable au stockage des clés d'accès au sein de l'instance EC2. Pour attribuer un AWS rôle à une instance EC2 et le mettre à la disposition de toutes ses applications, vous devez créer un profil d'instance attaché à l'instance. Un profil d'instance contient le rôle et permet aux programmes qui s'exécutent sur l'instance EC2 d'obtenir des informations d'identification temporaires. Pour plus d'informations, consultez Utilisation d'un rôle IAM pour accorder des autorisations à des applications s'exécutant sur des instances Amazon EC2 dans le Guide de l'utilisateur IAM.

Pour savoir dans quel cas utiliser des rôles ou des utilisateurs IAM, consultez Quand créer un rôle IAM (au lieu d'un utilisateur) dans le Guide de l'utilisateur IAM.

Gestion des accès à l’aide de politiques

Vous contrôlez l'accès en AWS créant des politiques et en les associant à AWS des identités ou à des ressources. Une politique est un objet AWS qui, lorsqu'il est associé à une identité ou à une ressource, définit leurs autorisations. AWS évalue ces politiques lorsqu'un principal (utilisateur, utilisateur root ou session de rôle) fait une demande. Les autorisations dans les politiques déterminent si la demande est autorisée ou refusée. La plupart des politiques sont stockées AWS sous forme de documents JSON. Pour plus d’informations sur la structure et le contenu des documents de politique JSON, consultez Vue d’ensemble des politiques JSON dans le Guide de l’utilisateur IAM.

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C’est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

Par défaut, les utilisateurs et les rôles ne disposent d’aucune autorisation. Pour octroyer aux utilisateurs des autorisations d’effectuer des actions sur les ressources dont ils ont besoin, un administrateur IAM peut créer des politiques IAM. L’administrateur peut ensuite ajouter les politiques IAM aux rôles et les utilisateurs peuvent assumer les rôles.

Les politiques IAM définissent les autorisations d’une action, quelle que soit la méthode que vous utilisez pour exécuter l’opération. Par exemple, supposons que vous disposiez d’une politique qui autorise l’action `iam:GetRole`. Un utilisateur appliquant cette politique peut obtenir des informations sur le rôle à partir de AWS Management Console AWS CLI, de ou de l' AWS API.

Politiques basées sur l’identité

Les politiques basées sur l’identité sont des documents de politique d’autorisations JSON que vous pouvez attacher à une identité telle qu’un utilisateur, un groupe d’utilisateurs ou un rôle IAM. Ces politiques contrôlent quel type d’actions des utilisateurs et des rôles peuvent exécuter, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l’identité, consultez Création de politiques IAM dans le Guide de l’utilisateur IAM.

Les politiques basées sur l’identité peuvent être classées comme des politiques en ligne ou des politiques gérées. Les politiques en ligne sont intégrées directement à un utilisateur, groupe ou rôle. Les politiques gérées sont des politiques autonomes que vous pouvez associer à plusieurs utilisateurs, groupes et rôles au sein de votre Compte AWS. Les politiques gérées incluent les politiques AWS gérées et les politiques gérées par le client. Pour découvrir comment choisir entre

une politique gérée et une politique en ligne, consultez Choix entre les politiques gérées et les politiques en ligne dans le Guide de l'utilisateur IAM.

politiques basées sur les ressources

Les politiques basées sur les ressources sont des documents de politique JSON que vous attachez à une ressource. Des politiques basées sur les ressources sont, par exemple, les politiques de confiance de rôle IAM et des politiques de compartiment. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l'accès à une ressource spécifique. Pour la ressource dans laquelle se trouve la politique, cette dernière définit quel type d'actions un principal spécifié peut effectuer sur cette ressource et dans quelles conditions. Vous devez spécifier un principal dans une politique basée sur les ressources. Les principaux peuvent inclure des comptes, des utilisateurs, des rôles, des utilisateurs fédérés ou. Services AWS

Les politiques basées sur les ressources sont des politiques en ligne situées dans ce service. Vous ne pouvez pas utiliser les politiques AWS gérées par IAM dans une stratégie basée sur les ressources.

Listes de contrôle d'accès (ACL)

Les listes de contrôle d'accès (ACL) vérifie quels principals (membres de compte, utilisateurs ou rôles) ont l'autorisation d'accéder à une ressource. Les listes de contrôle d'accès sont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format de document de politique JSON.

Amazon S3 et Amazon VPC sont des exemples de services qui prennent en charge les ACL. AWS WAF Pour en savoir plus sur les listes de contrôle d'accès, consultez Vue d'ensemble des listes de contrôle d'accès (ACL) dans le Guide du développeur Amazon Simple Storage Service.

Autres types de politique

AWS prend en charge d'autres types de politiques moins courants. Ces types de politiques peuvent définir le nombre maximum d'autorisations qui vous sont accordées par des types de politiques plus courants.

- Limite d'autorisations : une limite d'autorisations est une fonctionnalité avancée dans laquelle vous définissez le nombre maximal d'autorisations qu'une politique basée sur l'identité peut accorder à une entité IAM (utilisateur ou rôle IAM). Vous pouvez définir une limite d'autorisations pour

une entité. Les autorisations en résultant représentent la combinaison des politiques basées sur l'identité d'une entité et de ses limites d'autorisation. Les politiques basées sur les ressources qui spécifient l'utilisateur ou le rôle dans le champ `Principal` ne sont pas limitées par les limites d'autorisations. Un refus explicite dans l'une de ces politiques remplace l'autorisation. Pour plus d'informations sur les limites d'autorisations, consultez Limites d'autorisations pour des entités IAM dans le Guide de l'utilisateur IAM.

- Politiques de contrôle des services (SCP) — Les SCP sont des politiques JSON qui spécifient les autorisations maximales pour une organisation ou une unité organisationnelle (UO) dans. AWS Organizations AWS Organizations est un service permettant de regrouper et de gérer de manière centralisée Comptes AWS les multiples propriétés de votre entreprise. Si vous activez toutes les fonctionnalités d'une organisation, vous pouvez appliquer les politiques de contrôle des services (SCP) à l'un ou à l'ensemble de vos comptes. Le SCP limite les autorisations pour les entités figurant dans les comptes des membres, y compris chacune Utilisateur racine d'un compte AWS d'entre elles. Pour plus d'informations sur les organisations et les SCP, consultez Fonctionnement des SCP dans le Guide de l'utilisateur AWS Organizations .
- Politiques de séance : les politiques de séance sont des politiques avancées que vous utilisez en tant que paramètre lorsque vous créez par programmation une séance temporaire pour un rôle ou un utilisateur fédéré. Les autorisations de séance en résultant sont une combinaison des politiques basées sur l'identité de l'utilisateur ou du rôle et des politiques de séance. Les autorisations peuvent également provenir d'une politique basée sur les ressources. Un refus explicite dans l'une de ces politiques annule l'autorisation. Pour plus d'informations, consultez politiques de séance dans le Guide de l'utilisateur IAM.

Plusieurs types de politique

Lorsque plusieurs types de politiques s'appliquent à la requête, les autorisations en résultant sont plus compliquées à comprendre. Pour savoir comment AWS déterminer s'il faut autoriser une demande lorsque plusieurs types de politiques sont impliqués, consultez la section Logique d'évaluation des politiques dans le guide de l'utilisateur IAM.

Comment Amazon QLDB fonctionne avec IAM

Avant d'utiliser IAM pour gérer l'accès à QLDB, découvrez quelles fonctionnalités IAM peuvent être utilisées avec QLDB.

Fonctionnalités IAM que vous pouvez utiliser avec Amazon QLDB

Fonction IAM	Prise en charge de QLDB
Politiques basées sur l'identité	Oui
Politiques basées sur les ressources	Non
Actions de politique	Oui
Ressources de politique	Oui
Clés de condition d'une politique	Oui
ACL	Non
ABAC (étiquettes dans les politiques)	Oui
Informations d'identification temporaires	Oui
Autorisations de principals	Non
Fonctions de service	Oui
Rôles liés à un service	Non

Pour obtenir une vue d'ensemble de la façon dont QLDB et les Services AWS autres fonctionnalités fonctionnent avec la plupart des fonctionnalités IAM, Services AWS consultez le guide de l'utilisateur IAM consacré à leur fonctionnement avec IAM.

Politiques basées sur l'identité pour QLDB

Prend en charge les politiques basées sur l'identité	Oui

Les politiques basées sur l'identité sont des documents de politique d'autorisations JSON que vous pouvez attacher à une identité telle qu'un utilisateur, un groupe d'utilisateurs ou un rôle IAM. Ces politiques contrôlent quel type d'actions des utilisateurs et des rôles peuvent exécuter, sur quelles

ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l'identité, consultez Création de politiques IAM dans le Guide de l'utilisateur IAM.

Avec les politiques IAM basées sur l'identité, vous pouvez spécifier des actions et ressources autorisées ou refusées, ainsi que les conditions dans lesquelles les actions sont autorisées ou refusées. Vous ne pouvez pas spécifier le principal dans une politique basée sur une identité car celle-ci s'applique à l'utilisateur ou au rôle auquel elle est attachée. Pour découvrir tous les éléments que vous utilisez dans une politique JSON, consultez Références des éléments de politique JSON IAM dans le Guide de l'utilisateur IAM.

Exemples de politiques basées sur l'identité pour QLDB

Pour consulter des exemples de politiques basées sur l'identité QLDB, consultez. Exemples de politiques basées sur l'identité pour Amazon QLDB

Politiques basées sur les ressources au sein de QLDB

Prend en charge les politiques basées sur les ressources	Non

Les politiques basées sur les ressources sont des documents de politique JSON que vous attachez à une ressource. Des politiques basées sur les ressources sont, par exemple, les politiques de confiance de rôle IAM et des politiques de compartiment. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l'accès à une ressource spécifique. Pour la ressource dans laquelle se trouve la politique, cette dernière définit quel type d'actions un principal spécifié peut effectuer sur cette ressource et dans quelles conditions. Vous devez spécifier un principal dans une politique basée sur les ressources. Les principaux peuvent inclure des comptes, des utilisateurs, des rôles, des utilisateurs fédérés ou. Services AWS

Pour permettre un accès intercompte, vous pouvez spécifier un compte entier ou des entités IAM dans un autre compte en tant que principal dans une politique basée sur les ressources. L'ajout d'un principal entre comptes à une politique basée sur les ressources ne représente qu'une partie de l'instauration de la relation d'approbation. Lorsque le principal et la ressource sont différents Comptes AWS, un administrateur IAM du compte sécurisé doit également accorder à l'entité principale (utilisateur ou rôle) l'autorisation d'accéder à la ressource. Pour ce faire, il attache une politique basée sur une identité à l'entité. Toutefois, si une politique basée sur des ressources

accorde l'accès à un principal dans le même compte, aucune autre politique basée sur l'identité n'est requise. Pour plus d'informations, consultez Différence entre les rôles IAM et les politiques basées sur une ressource dans le Guide de l'utilisateur IAM.

Actions politiques pour QLDB

Prend en charge les actions de politique	Oui

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Action` d'une politique JSON décrit les actions que vous pouvez utiliser pour autoriser ou refuser l'accès à une politique. Les actions de stratégie portent généralement le même nom que l'opération AWS d'API associée. Il existe quelques exceptions, telles que les actions avec autorisations uniquement qui n'ont pas d'opération API correspondante. Certaines opérations nécessitent également plusieurs actions dans une politique. Ces actions supplémentaires sont nommées actions dépendantes.

Intégration d'actions dans une stratégie afin d'accorder l'autorisation d'exécuter les opérations associées.

Pour consulter la liste des actions QLDB, consultez la section Actions définies par Amazon QLDB dans le Service Authorization Reference.

Les actions de stratégie dans QLDB utilisent le préfixe suivant avant l'action :

```
qldb
```

Pour indiquer plusieurs actions dans une seule déclaration, séparez-les par des virgules.

```
"Action": [
      "qldb:action1",
      "qldb:action2"
]
```

Vous pouvez aussi spécifier plusieurs actions à l'aide de caractères génériques (*). Par exemple, pour spécifier toutes les actions qui commencent par le mot `Describe`, incluez l'action suivante :

```
"Action": "qldb:Describe*"
```

Pour interagir avec l'API de données transactionnelles QLDB (session QLDB) en exécutant des instructions partiQL sur un registre, vous devez autoriser l'action comme suit. `SendCommand`

```
"Action": "qldb:SendCommand"
```

Pour les registres en mode `STANDARD` autorisations, consultez le Référence des autorisations partiQL pour connaître les autorisations supplémentaires requises pour chaque commande partiQL.

Pour consulter des exemples de politiques basées sur l'identité QLDB, consultez. Exemples de politiques basées sur l'identité pour Amazon QLDB

Ressources relatives aux politiques pour QLDB

Prend en charge les ressources de politique	Oui

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément de politique JSON `Resource` indique le ou les objets auxquels l'action s'applique. Les instructions doivent inclure un élément `Resource` ou `NotResource`. Il est recommandé de définir une ressource à l'aide de son Amazon Resource Name (ARN). Vous pouvez le faire pour des actions qui prennent en charge un type de ressource spécifique, connu sous la dénomination autorisations de niveau ressource.

Pour les actions qui ne sont pas compatibles avec les autorisations de niveau ressource, telles que les opérations de liste, utilisez un caractère générique (*) afin d'indiquer que l'instruction s'applique à toutes les ressources.

```
"Resource": "*"
```

Pour consulter la liste des types de ressources QLDB et de leurs ARN, consultez la section Ressources définies par Amazon QLDB dans le Service Authorization Reference. Pour savoir avec

quelles actions vous pouvez spécifier l'ARN de chaque ressource, consultez Actions définies par Amazon QLDB.

Dans QLDB, les ressources principales sont les registres. QLDB prend également en charge d'autres types de ressources : tables et flux. Toutefois, vous ne pouvez créer des tables et des flux que dans le contexte d'un registre existant.

Une table QLDB est une vue matérialisée d'un ensemble non ordonné de révisions de documents provenant du journal du grand livre. Dans le mode `STANDARD` autorisations d'un registre, vous devez créer des politiques IAM qui accordent des autorisations pour exécuter des instructions partiQL sur cette ressource de table. Avec des autorisations sur une ressource de table, vous pouvez exécuter des instructions qui accèdent à l'état actuel de la table. Vous pouvez également consulter l'historique des révisions de la table à l'aide de la `history()` fonction intégrée. Pour en savoir plus, veuillez consulter la section Commencer à utiliser le mode d'autorisation standard dans Amazon QLDB.

> **Note**
>
> L'`CREATE TABLE`instruction crée une table avec un identifiant unique et le nom de table fourni. Le nom de table fourni doit être unique parmi toutes les tables actives. Cependant, QLDB vous permet de désactiver les tables. Il est donc possible que plusieurs tables inactives portent le même nom de table. Par conséquent, les ARN des ressources de table font référence à l'ID unique attribué par le système plutôt qu'au nom de table défini par l'utilisateur.

Chaque registre fournit également une ressource de catalogue définie par le système que vous pouvez interroger pour répertorier toutes les tables et tous les index d'un registre. Pour plus d'informations sur le modèle d'objet de données QLDB, consultez. Concepts et terminologie de base d'Amazon QLDB

Ces ressources sont associées à des ARN uniques, comme indiqué dans le tableau suivant.

Type de ressource	ARN
ledger	`arn:${Partition}:qldb:${Region}:${Account}:ledger/${LedgerName}`

Type de ressource	ARN
table	`arn:${Partition}:qldb:${Region}:${Account}:ledger/${LedgerName}/table/${TableId}`
catalog	`arn:${Partition}:qldb:${Region}:${Account}:ledger/${LedgerName}/information_schema/user_tables`
stream	`arn:${Partition}:qldb:${Region}:${Account}:stream/${LedgerName}/${StreamId}`

Par exemple, pour spécifier la `myExampleLedger` ressource dans votre relevé, utilisez l'ARN suivant.

```
"Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
```

Pour spécifier plusieurs ressources dans une seule instruction, séparez leurs ARN par des virgules.

```
"Resource": [
      "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger1",
      "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger2"
]
```

Pour consulter des exemples de politiques basées sur l'identité QLDB, consultez. Exemples de politiques basées sur l'identité pour Amazon QLDB

Clés de conditions de politique pour QLDB

Prend en charge les clés de condition de politique spécifiques au service	Oui

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Condition` (ou le bloc `Condition`) vous permet de spécifier des conditions lorsqu'une instruction est appliquée. L'élément `Condition` est facultatif. Vous pouvez créer des expressions conditionnelles qui utilisent des opérateurs de condition, tels que les signes égal ou inférieur à, pour faire correspondre la condition de la politique aux valeurs de la demande.

Si vous spécifiez plusieurs éléments `Condition` dans une instruction, ou plusieurs clés dans un seul élément `Condition`, AWS les évalue à l'aide d'une opération `AND` logique. Si vous spécifiez plusieurs valeurs pour une seule clé de condition, AWS évalue la condition à l'aide d'une `OR` opération logique. Toutes les conditions doivent être remplies avant que les autorisations associées à l'instruction ne soient accordées.

Vous pouvez aussi utiliser des variables d'espace réservé quand vous spécifiez des conditions. Par exemple, vous pouvez accorder à un utilisateur IAM l'autorisation d'accéder à une ressource uniquement si elle est balisée avec son nom d'utilisateur IAM. Pour plus d'informations, consultez Éléments d'une politique IAM : variables et identifications dans le Guide de l'utilisateur IAM.

AWS prend en charge les clés de condition globales et les clés de condition spécifiques au service. Pour voir toutes les clés de condition AWS globales, voir les clés de contexte de condition AWS globales dans le guide de l'utilisateur IAM.

Pour consulter la liste des clés de condition QLDB, consultez la section Clés de condition pour Amazon QLDB dans la référence d'autorisation de service. Pour savoir avec quelles actions et ressources vous pouvez utiliser une clé de condition, consultez Actions définies par Amazon QLDB.

Les `PartiQLDropTable` actions `PartiQLDropIndex` et prennent en charge la clé de `qldb:Purge` condition. Cette clé de condition filtre l'accès en fonction de la `purge` valeur spécifiée dans une instruction partiQLDROP. Cependant, QLDB `purge = true` ne prend actuellement en charge que les instructions, `purge = false` et `DROP INDEX` pour les instructions. `DROP TABLE` D'autres actions QLDB prennent en charge certaines clés de condition globales.

Pour consulter des exemples de politiques basées sur l'identité QLDB, consultez. Exemples de politiques basées sur l'identité pour Amazon QLDB

Listes de contrôle d'accès (ACL) dans QLDB

Prend en charge les listes ACL	Non

Les listes de contrôle d'accès (ACL) vérifient quels principaux (membres de compte, utilisateurs ou rôles) ont l'autorisation d'accéder à une ressource. Les listes de contrôle d'accès sont similaires aux

politiques basées sur les ressources, bien qu'elles n'utilisent pas le format de document de politique JSON.

Contrôle d'accès basé sur les attributs (ABAC) avec QLDB

Prend en charge ABAC (étiquettes dans les politiques)	Oui

Le contrôle d'accès par attributs (ABAC) est une stratégie d'autorisation qui définit des autorisations en fonction des attributs. Dans AWS, ces attributs sont appelés balises. Vous pouvez associer des balises aux entités IAM (utilisateurs ou rôles) et à de nombreuses AWS ressources. L'étiquetage des entités et des ressources est la première étape d'ABAC. Vous concevez ensuite des politiques ABAC pour autoriser des opérations quand l'identification du principal correspond à celle de la ressource à laquelle il tente d'accéder.

L'ABAC est utile dans les environnements qui connaissent une croissance rapide et pour les cas où la gestion des politiques devient fastidieuse.

Pour contrôler l'accès basé sur des étiquettes, vous devez fournir les informations d'étiquette dans l'élément de condition d'une politique utilisant les clés de condition `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` ou `aws:TagKeys`.

Si un service prend en charge les trois clés de condition pour tous les types de ressources, alors la valeur pour ce service est Oui. Si un service prend en charge les trois clés de condition pour certains types de ressources uniquement, la valeur est Partielle.

Pour plus d'informations sur l'ABAC, consultez Qu'est-ce que le contrôle d'accès basé sur les attributs (ABAC) ? dans le Guide de l'utilisateur IAM. Pour accéder à un didacticiel décrivant les étapes de configuration de l'ABAC, consultez Utilisation du contrôle d'accès par attributs (ABAC) dans le Guide de l'utilisateur IAM.

Pour plus d'informations sur le balisage des ressources QLDB, consultez. Balisage des ressources Amazon QLDB

Pour visualiser un exemple de politique basée sur l'identité permettant de limiter l'accès à une ressource en fonction des balises de cette ressource, consultez Mise à jour des registres QLDB en fonction des balises.

Utilisation d'informations d'identification temporaires avec QLDB

Prend en charge les informations d’identification temporaires	Oui

Certains Services AWS ne fonctionnent pas lorsque vous vous connectez à l'aide d'informations d'identification temporaires. Pour plus d'informations, y compris celles qui Services AWS fonctionnent avec des informations d'identification temporaires, consultez Services AWS la section relative à l'utilisation d'IAM dans le guide de l'utilisateur d'IAM.

Vous utilisez des informations d'identification temporaires si vous vous connectez à l' AWS Management Console aide d'une méthode autre qu'un nom d'utilisateur et un mot de passe. Par exemple, lorsque vous accédez à AWS l'aide du lien d'authentification unique (SSO) de votre entreprise, ce processus crée automatiquement des informations d'identification temporaires. Vous créez également automatiquement des informations d’identification temporaires lorsque vous vous connectez à la console en tant qu’utilisateur, puis changez de rôle. Pour plus d’informations sur le changement de rôle, consultez Changement de rôle (console) dans le Guide de l’utilisateur IAM.

Vous pouvez créer manuellement des informations d'identification temporaires à l'aide de l' AWS API AWS CLI or. Vous pouvez ensuite utiliser ces informations d'identification temporaires pour y accéder AWS. AWS recommande de générer dynamiquement des informations d'identification temporaires au lieu d'utiliser des clés d'accès à long terme. Pour plus d’informations, consultez Informations d’identification de sécurité temporaires dans IAM.

Autorisations principales interservices pour QLDB

Prend en charge les sessions d'accès direct (FAS)	Non

Lorsque vous utilisez un utilisateur ou un rôle IAM pour effectuer des actions AWS, vous êtes considéré comme un mandant. Lorsque vous utilisez certains services, vous pouvez effectuer une action qui initie une autre action dans un autre service. FAS utilise les autorisations du principal appelant et Service AWS, associées Service AWS à la demande, pour adresser des demandes aux services en aval. Les demandes FAS ne sont effectuées que lorsqu'un service reçoit une demande qui nécessite des interactions avec d'autres personnes Services AWS ou des ressources pour être traitée. Dans ce cas, vous devez disposer d’autorisations nécessaires pour effectuer les deux

actions. Pour plus de détails sur une politique lors de la formulation de demandes FAS, consultez Transmission des sessions d'accès.

Rôles de service pour QLDB

Prend en charge les fonctions du service	Oui

Une fonction de service est un rôle IAM qu'un service endosse pour accomplir des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer une fonction du service à partir d'IAM. Pour plus d'informations, consultez Création d'un rôle pour la délégation d'autorisations à un Service AWS dans le Guide de l'utilisateur IAM.

> **Warning**
>
> La modification des autorisations associées à un rôle de service peut interrompre les fonctionnalités de QLDB. Modifiez les rôles de service uniquement lorsque QLDB fournit des instructions à cet effet.

QLDB prend en charge les rôles de service pour `ExportJournalToS3` les opérations `StreamJournalToKinesis` et API, comme décrit dans la section suivante.

Choix d'un rôle IAM dans QLDB

Lorsque vous exportez ou diffusez des blocs de journal dans QLDB, vous devez choisir un rôle pour permettre à QLDB d'écrire des objets vers la destination donnée en votre nom. Si vous avez déjà créé un rôle de service, QLDB vous propose une liste de rôles parmi lesquels choisir. Il est important de choisir un rôle qui autorise l'accès à l'écriture dans le compartiment Amazon S3 que vous avez spécifié pour une exportation, ou à la ressource Amazon Kinesis Data Streams spécifiée pour un flux. Pour plus d'informations, consultez Autorisations d'exportation de journaux dans QLDB ou Autorisations de diffusion dans QLDB.

> **Note**
>
> Pour transmettre un rôle à QLDB lorsque vous demandez une exportation de journal ou un flux, vous devez disposer des autorisations nécessaires pour effectuer `iam:PassRole` l'action sur la ressource du rôle IAM. Cela s'ajoute aux autorisations

à effectuer `qldb:ExportJournalToS3` sur la ressource QLDB Ledger ou `qldb:StreamJournalToKinesis` sur la sous-ressource de flux QLDB.

Rôles liés à un service pour QLDB

Prend en charge les rôles liés à un service	Non

Un rôle lié à un service est un type de rôle de service lié à un. Service AWS Le service peut endosser le rôle afin d'effectuer une action en votre nom. Les rôles liés à un service apparaissent dans votre Compte AWS fichier et appartiennent au service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.

Pour plus de détails sur la création ou la gestion de rôles liés à un service, consultez la section relative à l'Services AWS utilisation d'IAM. Recherchez un service dans le tableau qui inclut un `Yes` dans la colonne Rôle lié à un service. Choisissez le lien Oui pour consulter la documentation du rôle lié à ce service.

Commencer à utiliser le mode d'autorisation standard dans Amazon QLDB

Utilisez cette section pour commencer à utiliser le mode d'autorisation standard dans Amazon QLDB. Cette section fournit un tableau de référence pour vous aider à écrire une politique basée sur l'identité dans AWS Identity and Access Management (IAM) pour les actions partiQL et les ressources de table dans QLDB. Il inclut également un step-by-step didacticiel pour créer des politiques d'autorisation dans IAM, ainsi que des instructions pour trouver un ARN de table et créer des balises de table dans QLDB.

Rubriques

- Le mode STANDARD autorisations
- Référence des autorisations partiQL
- Trouver un ID de table et un ARN
- Tableaux de balisage
- Tutoriel de démarrage rapide : Création de politiques d'autorisation

Le mode **`STANDARD`** autorisations

QLDB prend désormais en charge `STANDARD` un mode d'autorisations pour les ressources du registre. Le mode d'autorisations standard permet un contrôle d'accès avec une granularité plus fine pour les registres, les tables et les commandes partiQL. Par défaut, ce mode refuse toutes les demandes d'exécution de commandes partiQL sur les tables d'un registre.

> **Note**
>
> Auparavant, le seul mode d'autorisation disponible pour un registre était`ALLOW_ALL`. Ce `ALLOW_ALL` mode permet le contrôle d'accès avec une granularité au niveau de l'API pour les registres et continue d'être pris en charge, mais n'est pas recommandé, pour les registres QLDB. Ce mode permet aux utilisateurs autorisés par l'`SendCommand`API d'exécuter toutes les commandes partiQL sur toutes les tables du registre spécifié par la politique d'autorisation (d'où « autoriser toutes » les commandes partiQL).
> Vous pouvez modifier le mode d'autorisation des registres existants de `ALLOW_ALL` à`STANDARD`. Pour plus d'informations, veuillez consulter Migration vers le mode d'autorisations standard.

Pour autoriser les commandes en mode standard, vous devez créer une politique d'autorisation dans IAM pour des ressources de table et des actions partiQL spécifiques. Cela s'ajoute à l'autorisation `SendCommand` d'API pour le registre. Pour faciliter les politiques dans ce mode, QLDB a introduit un ensemble d'actions IAM pour les commandes partiQL et Amazon Resource Names (ARN) pour les tables QLDB. Pour plus d'informations sur le modèle d'objet de données QLDB, consultez. Concepts et terminologie de base d'Amazon QLDB

Référence des autorisations partiQL

Le tableau suivant répertorie chaque commande QLDB partiQL, les actions IAM correspondantes pour lesquelles vous devez accorder des autorisations pour exécuter la commande, AWS ainsi que les ressources pour lesquelles vous pouvez accorder ces autorisations. Vous spécifiez les actions dans le champ `Action` de la politique ainsi que la valeur des ressources dans le champ `Resource` de la politique.

Important

- Les politiques IAM qui accordent des autorisations à ces commandes partiQL ne s'appliquent à votre registre que si `STANDARD` le mode d'autorisations est attribué au registre. Ces politiques ne sont pas applicables aux registres en mode `ALLOW_ALL` autorisations.

  Pour savoir comment spécifier le mode d'autorisation lorsque vous créez ou mettez à jour un registreOpérations de base pour les registres Amazon QLDB, voir ou Étape 1 : Créer un nouveau registre dans Getting started with the console.
- Pour exécuter des commandes partiQL sur un registre, vous devez également autoriser l'action d'`SendCommand`API pour la ressource du registre. Cela s'ajoute aux actions partiQL et aux ressources de table répertoriées dans le tableau suivant. Pour plus d'informations, consultez Transactions de données en cours.

Commandes partiQL Amazon QLDB et autorisations requises

Command	Autorisations requises (actions IAM)	Ressources	Actions dépendantes
CREATE TABLE	`qldb:PartiQLCreateTable`	`arn:aws:qldb:region:account-id:ledger/ledger-name/table/*`	`qldb:TagResource` (pour le balisage lors de la création)
DROP TABLE	`qldb:PartiQLDropTable`	`arn:aws:qldb:region:account-id:ledger/ledger-name/table/table-id`	

Command	Autorisations requises (actions IAM)	Ressources	Actions dépendantes
TABLEAU DE DÉBALLAG	qldb:PartiQLUndropTable	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	
CREATE INDEX	qldb:PartiQLCreateIndex	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	
DROP INDEX	qldb:PartiQLDropIndex	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	
DELETE FROM-REMOVE (pour les documents entiers)	qldb:PartiQLDelete	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	qldb:PartiQLSelect
INSERT	qldb:PartiQLInsert	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	
MISE A JOUR	qldb:PartiQLUpdate	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	qldb:PartiQLSelect
DE (INSÉRER, SUPPRIME ou DÉFINIR)			

Command	Autorisations requises (actions IAM)	Ressources	Actions dépendantes
REDACT_REVISION(proc stockée)	qldb:PartiQLRedact	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	
SÉLECTIONNEZ À PARTIR DE TABLE_NAME	qldb:PartiQLSelect	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	
SÉLECTIONNEZ PARMI information_schema.user_tables	qldb:PartiQLSelect	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /information_schema/user_tables	
SÉLECTIONNEZ DANS L'historique (table_name)	qldb:PartiQLHistoryFunction	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	

Pour des exemples de documents de politique IAM qui accordent des autorisations à ces commandes partiQL, passez Tutoriel de démarrage rapide : Création de politiques d'autorisation à ou consultez. Exemples de politiques basées sur l'identité pour Amazon QLDB

Trouver un ID de table et un ARN

Vous pouvez trouver un identifiant de table en utilisant AWS Management Console ou en interrogeant la table information_schema.user_tables. Pour afficher les détails de la table sur la console ou pour interroger cette table du catalogue système, vous devez avoir l'`SELECT`autorisation d'accéder à la ressource du catalogue système. Par exemple, pour trouver l'ID de la `Vehicle` table, vous pouvez exécuter l'instruction suivante.

```
SELECT * FROM information_schema.user_tables
WHERE name = 'Vehicle'
```

Cette requête renvoie les résultats dans un format similaire à celui de l'exemple suivant.

```
{
    tableId: "Au1EiThbt8s0z9wM26REZN",
    name: "Vehicle",
    indexes: [
        { indexId: "Djg2nt0yIs2GY0T29Kud1z", expr: "[VIN]", status: "ONLINE" },
        { indexId: "4tPW3fUhaVhDinRgKRLhGU", expr: "[LicensePlateNumber]", status:
 "BUILDING" }
    ],
    status: "ACTIVE"
}
```

Pour accorder l'autorisation d'exécuter des instructions partiQL sur une table, vous devez spécifier une ressource de table au format ARN suivant.

```
arn:aws:qldb:${region}:${account-id}:ledger/${ledger-name}/table/${table-id}
```

Voici un exemple d'ARN de table pour l'ID de table`Au1EiThbt8s0z9wM26REZN`.

```
arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/Au1EiThbt8s0z9wM26REZN
```

Utilisation de la console

Vous pouvez également utiliser la console QLDB pour rechercher l'ARN d'une table.

Pour trouver l'ARN d'une table (console)

1. Connectez-vous à la AWS Management Console console Amazon QLDB et ouvrez-la à l'adresse https://console.aws.amazon.com/qldb.

2. Dans le volet de navigation, choisissez Ledgers.
3. Dans la liste des registres, choisissez le nom du registre dont vous souhaitez rechercher l'ARN de la table.
4. Sur la page des détails du registre, sous l'onglet Tables, recherchez le nom de la table dont vous souhaitez rechercher l'ARN. Pour copier l'ARN, cliquez sur l'icône de copie () à côté de celui-ci.

Tableaux de balisage

Vous pouvez baliser les ressources de votre table. Pour gérer les balises pour les tables existantes, utilisez les opérations AWS Management Console ou l'API `TagResourceUntagResource`, et`ListTagsForResource`. Pour plus d'informations, consultez Balisage des ressources Amazon QLDB.

> **Note**
>
> Les ressources de table n'héritent pas des balises de leur ressource de registre racine. Le balisage des tables lors de leur création est actuellement pris en charge pour les registres en mode `STANDARD` autorisations uniquement.

Vous pouvez également définir des balises de table lors de la création de la table à l'aide de la console QLDB ou en les spécifiant dans une `CREATE TABLE` instruction partiQL. L'exemple suivant crée une table nommée `Vehicle` avec la balise`environment=production`.

```
CREATE TABLE Vehicle WITH (aws_tags = `{'environment': 'production'}`)
```

Le balisage des tables lors de leur création nécessite l'accès aux `qldb:TagResource` actions `qldb:PartiQLCreateTable` et.

En attribuant des étiquettes aux ressources au moment de la création, vous pouvez supprimer la nécessité d'exécuter des scripts d'identification personnalisés après la création de ressources. Une fois qu'une table est balisée, vous pouvez contrôler l'accès à la table en fonction de ces balises. Par exemple, vous pouvez accorder un accès complet uniquement aux tables dotées d'une balise spécifique. Pour un exemple de politique JSON, consultezAccès complet à toutes les actions en fonction des balises du tableau.

Utilisation de la console

Vous pouvez également utiliser la console QLDB pour définir des balises de table lors de la création de la table.

Pour étiqueter un tableau lors de sa création (console)

1. Connectez-vous à la AWS Management Console console Amazon QLDB et ouvrez-la à l'adresse https://console.aws.amazon.com/qldb.
2. Dans le volet de navigation, choisissez Ledgers.
3. Dans la liste des livres, choisissez le nom du livre dans lequel vous souhaitez créer la table.
4. Sur la page des détails du registre, sous l'onglet Tables, choisissez Créer une table.
5. Sur la page Créer une table, procédez comme suit :

   - Nom de la table — Entrez un nom de table.
   - Balises : ajoutez des métadonnées à la table en attachant des balises sous forme de paires clé-valeur. Vous pouvez ajouter des balises à votre tableau pour mieux les organiser et les identifier.

     Choisissez Ajouter une balise, puis entrez les paires clé-valeur appropriées.
6. Lorsque les paramètres vous conviennent, choisissez Créer une table.

Tutoriel de démarrage rapide : Création de politiques d'autorisation

Ce didacticiel explique les étapes à suivre pour créer des politiques d'autorisations dans IAM pour un registre Amazon QLDB en mode autorisations. `STANDARD` Vous pouvez ensuite attribuer les autorisations à vos utilisateurs, groupes ou rôles.

Pour plus d'exemples de documents de politique IAM qui accordent des autorisations aux commandes partiQL et aux ressources de table, consultez. Exemples de politiques basées sur l'identité pour Amazon QLDB

Rubriques

- Prérequis
- Création d'une politique en lecture seule
- Création d'une politique d'accès complet

- Création d'une politique de lecture seule pour une table spécifique
- Attribution d’autorisations

Prérequis

Avant de commencer, assurez-vous d'effectuer les opérations suivantes :

1. Suivez les instructions de AWS configuration indiquées dansAccès à Amazon QLDB, si ce n'est pas déjà fait. Ces étapes incluent l'inscription AWS et la création d'un utilisateur administratif.
2. Créez un nouveau registre et choisissez le mode `STANDARD` d'autorisations pour le registre. Pour savoir comment procéder, consultez Étape 1 : Créer un nouveau registre la section Prise en main de la console, ouOpérations de base pour les registres Amazon QLDB.

Création d'une politique en lecture seule

Pour utiliser l'éditeur de stratégie JSON afin de créer une politique en lecture seule pour toutes les tables d'un registre en mode d'autorisations standard, procédez comme suit :

1. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/iam/.
2. Dans la colonne de navigation de gauche, sélectionnez Politiques.

   Si vous choisissez Politiques pour la première fois, la page Bienvenue dans les politiques gérées s'affiche. Sélectionnez Mise en route.
3. En haut de la page, sélectionnez Créer une politique.
4. Sélectionnez l’onglet JSON.
5. Copiez et collez le document de politique JSON suivant. Cet exemple de politique accorde un accès en lecture seule à toutes les tables d'un registre.

   Pour utiliser cette politique, remplacez *`us-east-1`*, 123456789012*`, et`* dans l'exemple par vos propres informations. *`myExampleLedger`*

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "QLDBSendCommandPermission",
      "Effect": "Allow",
```

```
      "Action": "qldb:SendCommand",
      "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
    },
    {
      "Sid": "QLDBPartiQLReadOnlyPermissions",
      "Effect": "Allow",
      "Action": [
        "qldb:PartiQLSelect",
        "qldb:PartiQLHistoryFunction"
      ],
      "Resource": [
        "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/*",
        "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/
information_schema/user_tables"
      ]
    }
  ]
}
```

6. Choisissez Examiner une politique.

> **Note**
>
> Vous pouvez basculer à tout moment entre les onglets Éditeur visuel et JSON. Toutefois, si vous apportez des modifications ou sélectionnez Examiner une politique dans l'onglet Éditeur visuel, IAM peut restructurer votre politique pour optimiser son affichage dans l'éditeur visuel. Pour de plus amples informations, consulte Restructuration d'une politique dans le Guide de l'utilisateur IAM.

7. Dans la page Examiner une politique, entrez un nom et éventuellement une description pour la politique que vous êtes en train de créer. Vérifiez le récapitulatif de la politique pour voir les autorisations accordées par votre politique. Sélectionnez ensuite Créer une politique pour enregistrer votre travail.

Création d'une politique d'accès complet

Pour créer une politique d'accès complet pour toutes les tables d'un registre QLDB en mode d'autorisations standard, procédez comme suit :

- Répétez les étapes précédentes à l'aide du document de politique suivant. Cet exemple de politique accorde l'accès à toutes les commandes partiQL pour toutes les tables d'un registre,

en utilisant des caractères génériques (*) pour couvrir toutes les actions partiQL et toutes les ressources d'un registre.

> **Warning**
>
> Il s'agit d'un exemple d'utilisation d'un caractère générique (*) pour autoriser toutes les actions PartiQL, y compris les opérations administratives et de lecture/écriture sur toutes les tables d'un registre QLDB. Il est préférable de spécifier explicitement chaque action à accorder, et uniquement ce dont cet utilisateur, ce rôle ou ce groupe a besoin.

Pour utiliser cette politique, remplacez *us-east-1*, 123456789012*, et* dans l'exemple par vos propres informations. *myExampleLedger*

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "QLDBSendCommandPermission",
      "Effect": "Allow",
      "Action": "qldb:SendCommand",
      "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
    },
    {
      "Sid": "QLDBPartiQLFullPermissions",
      "Effect": "Allow",
      "Action": [
        "qldb:PartiQL*"
      ],
      "Resource": [
        "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/*",
        "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/
information_schema/user_tables"
      ]
    }
  ]
}
```

Création d'une politique de lecture seule pour une table spécifique

Pour créer une politique d'accès en lecture seule pour une table spécifique dans un registre QLDB en mode d'autorisations standard, procédez comme suit :

1. Trouvez l'ARN de la table en utilisant AWS Management Console ou en interrogeant la table `information_schema.user_tables` du catalogue du système. Pour obtenir des instructions, veuillez consulter Trouver un ID de table et un ARN.
2. Utilisez l'ARN de la table pour créer une politique qui autorise l'accès en lecture seule à la table. Pour ce faire, répétez les étapes précédentes à l'aide du document de politique suivant.

   Cet exemple de politique accorde un accès en lecture seule à la table spécifiée uniquement. Dans cet exemple, l'ID de table est`Au1EiThbt8s0z9wM26REZN`. *Pour utiliser cette politique, remplacez us-east-1, 123456789012 et Au1 8S0z9WM26REZn dans l'exemple par vos myExampleLedgerpropres informations. EiThbt*

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "QLDBSendCommandPermission",
      "Effect": "Allow",
      "Action": "qldb:SendCommand",
      "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
    },
    {
      "Sid": "QLDBPartiQLReadOnlyPermissions",
      "Effect": "Allow",
      "Action": [
        "qldb:PartiQLSelect",
        "qldb:PartiQLHistoryFunction"
      ],
      "Resource": [
        "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/
table/Au1EiThbt8s0z9wM26REZN"
      ]
    }
  ]
}
```

Attribution d’autorisations

Après avoir créé une politique d'autorisations QLDB, vous attribuez les autorisations comme suit.

Pour activer l’accès, ajoutez des autorisations à vos utilisateurs, groupes ou rôles :

- Utilisateurs et groupes dans AWS IAM Identity Center :

  Créez un jeu d’autorisations. Suivez les instructions de la rubrique Création d’un jeu d’autorisations du Guide de l’utilisateur AWS IAM Identity Center .
- Utilisateurs gérés dans IAM par un fournisseur d’identité :

  Créez un rôle pour la fédération d’identité. Pour plus d’informations, voir la rubrique Création d’un rôle pour un fournisseur d’identité tiers (fédération) du Guide de l’utilisateur IAM.
- Utilisateurs IAM :
  - Créez un rôle que votre utilisateur peut assumer. Suivez les instructions de la rubrique Création d’un rôle pour un utilisateur IAM du Guide de l’utilisateur IAM.
  - (Non recommandé) Attachez une politique directement à un utilisateur ou ajoutez un utilisateur à un groupe d’utilisateurs. Suivez les instructions de la rubrique Ajout d’autorisations à un utilisateur (console) du Guide de l’utilisateur IAM.

Exemples de politiques basées sur l'identité pour Amazon QLDB

Par défaut, les utilisateurs et les rôles ne sont pas autorisés à créer ou à modifier des ressources QLDB. Ils ne peuvent pas non plus effectuer de tâches à l'aide de l'API AWS Management Console, AWS Command Line Interface (AWS CLI) ou de AWS l'API. Pour octroyer aux utilisateurs des autorisations d’effectuer des actions sur les ressources dont ils ont besoin, un administrateur IAM peut créer des politiques IAM. L’administrateur peut ensuite ajouter les politiques IAM aux rôles et les utilisateurs peuvent assumer les rôles.

Pour apprendre à créer une politique basée sur l’identité IAM à l’aide de ces exemples de documents de politique JSON, consultez Création de politiques dans l’onglet JSON dans le Guide de l’utilisateur IAM.

Pour plus de détails sur les actions et les types de ressources définis par QLDB, y compris le format des ARN pour chacun des types de ressources, consultez la section Actions, ressources et clés de condition pour Amazon QLDB dans le Service Authorization Reference.

Table des matières

Bonnes pratiques en matière de politiques

Les politiques basées sur l'identité déterminent si quelqu'un peut créer, accéder ou supprimer des ressources QLDB dans votre compte. Ces actions peuvent entraîner des frais pour votre Compte AWS. Lorsque vous créez ou modifiez des politiques basées sur l’identité, suivez ces instructions et recommandations :

- Commencez AWS par les politiques gérées et passez aux autorisations du moindre privilège : pour commencer à accorder des autorisations à vos utilisateurs et à vos charges de travail, utilisez les politiques AWS gérées qui accordent des autorisations pour de nombreux cas d'utilisation courants. Ils sont disponibles dans votre Compte AWS. Nous vous recommandons de réduire davantage les autorisations en définissant des politiques gérées par les AWS clients spécifiques à vos cas d'utilisation. Pour plus d’informations, consultez politiques gérées par AWS ou politiques gérées par AWS pour les activités professionnelles dans le Guide de l’utilisateur IAM.

- Accorder les autorisations de moindre privilège : lorsque vous définissez des autorisations avec des politiques IAM, accordez uniquement les autorisations nécessaires à l'exécution d'une seule tâche. Pour ce faire, vous définissez les actions qui peuvent être entreprises sur des ressources spécifiques dans des conditions spécifiques, également appelées autorisations de moindre privilège. Pour plus d'informations sur l'utilisation de IAM pour appliquer des autorisations, consultez politiques et autorisations dans IAM dans le Guide de l'utilisateur IAM.
- Utiliser des conditions dans les politiques IAM pour restreindre davantage l'accès : vous pouvez ajouter une condition à vos politiques afin de limiter l'accès aux actions et aux ressources. Par exemple, vous pouvez écrire une condition de politique pour spécifier que toutes les demandes doivent être envoyées via SSL. Vous pouvez également utiliser des conditions pour accorder l'accès aux actions de service si elles sont utilisées par le biais d'un service spécifique Service AWS, tel que AWS CloudFormation. Pour plus d'informations, consultez Conditions pour éléments de politique JSON IAM dans le Guide de l'utilisateur IAM.
- Utilisez IAM Access Analyzer pour valider vos politiques IAM afin de garantir des autorisations sécurisées et fonctionnelles : IAM Access Analyzer valide les politiques nouvelles et existantes de manière à ce que les politiques IAM respectent le langage de politique IAM (JSON) et les bonnes pratiques IAM. IAM Access Analyzer fournit plus de 100 vérifications de politiques et des recommandations exploitables pour vous aider à créer des politiques sécurisées et fonctionnelles. Pour plus d'informations, consultez Validation de politique IAM Access Analyzer dans le Guide de l'utilisateur IAM.
- Exiger l'authentification multifactorielle (MFA) : si vous avez un scénario qui nécessite des utilisateurs IAM ou un utilisateur root, activez l'authentification MFA pour une sécurité accrue. Compte AWS Pour exiger le MFA lorsque des opérations d'API sont appelées, ajoutez des conditions MFA à vos politiques. Pour plus d'informations, consultez Configuration de l'accès aux API protégé par MFA dans le Guide de l'utilisateur IAM.

Pour plus d'informations sur les bonnes pratiques dans IAM, consultez Bonnes pratiques de sécurité dans IAM dans le Guide de l'utilisateur IAM.

Utilisation de la console QLDB

Pour accéder à la console Amazon QLDB, vous devez disposer d'un ensemble minimal d'autorisations. Ces autorisations doivent vous permettre de répertorier et d'afficher les détails relatifs aux ressources QLDB de votre Compte AWS Si vous créez une stratégie basée sur l'identité qui est plus restrictive que l'ensemble minimum d'autorisations requis, la console ne fonctionnera pas comme prévu pour les entités (utilisateurs ou rôles) tributaires de cette stratégie.

Il n'est pas nécessaire d'accorder des autorisations de console minimales aux utilisateurs qui appellent uniquement l'API AWS CLI ou l' AWS API. Autorisez plutôt l'accès à uniquement aux actions qui correspondent à l'opération d'API qu'ils tentent d'effectuer.

Pour garantir aux utilisateurs et aux rôles un accès complet à la console QLDB et à toutes ses fonctionnalités, associez la politique gérée AWS suivante aux entités. Pour plus d'informationsAWS politiques gérées pour Amazon QLDB, consultez la section « Ajouter des autorisations à un utilisateur » dans le guide de l'utilisateur IAM.

```
AmazonQLDBConsoleFullAccess
```

Autorisations d'historique des requêtes

Outre les autorisations QLDB, certaines fonctionnalités de console nécessitent des autorisations pour le service de métadonnées de requête de base de données (préfixe de service :). `dbqms` Il s'agit d'un service interne uniquement qui gère vos requêtes récentes et enregistrées dans l'éditeur de requêtes de console pour QLDB et autres. Services AWS Pour une liste complète des actions de l'API DBQMS, voir Service de métadonnées de requête de base de données dans la référence d'autorisation de service.

Pour autoriser les autorisations relatives à l'historique des requêtes, vous pouvez utiliser la politique AWS gérée AmazonQLDB ConsoleFullAccess. Cette politique utilise un caractère générique (`dbqms:*`) pour autoriser toutes les actions DBQMS pour toutes les ressources.

Vous pouvez également créer une politique IAM personnalisée et inclure les actions DBQMS suivantes. L'éditeur de requêtes partiQL de la console QLDB nécessite des autorisations pour utiliser ces actions dans le cadre des fonctionnalités d'historique des requêtes.

```
dbqms:CreateFavoriteQuery
dbqms:CreateQueryHistory
dbqms:DeleteFavoriteQueries
dbqms:DeleteQueryHistory
dbqms:DescribeFavoriteQueries
dbqms:DescribeQueryHistory
dbqms:UpdateFavoriteQuery
```

Autorisations d'accès complet à la console sans historique des requêtes

Pour permettre un accès complet à la console QLDB sans aucune autorisation d'historique des requêtes, vous pouvez créer une politique IAM personnalisée qui exclut toutes les actions DBQMS.

Par exemple, le document de politique suivant autorise les mêmes autorisations que celles accordées par la politique AWS gérée AmazonQLDB ConsoleFullAccess, à l'exception des actions qui commencent par le préfixe de service. dbqms

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "qldb:CreateLedger",
        "qldb:UpdateLedger",
        "qldb:UpdateLedgerPermissionsMode",
        "qldb:DeleteLedger",
        "qldb:ListLedgers",
        "qldb:DescribeLedger",
        "qldb:ExportJournalToS3",
        "qldb:ListJournalS3Exports",
        "qldb:ListJournalS3ExportsForLedger",
        "qldb:DescribeJournalS3Export",
        "qldb:CancelJournalKinesisStream",
        "qldb:DescribeJournalKinesisStream",
        "qldb:ListJournalKinesisStreamsForLedger",
        "qldb:StreamJournalToKinesis",
        "qldb:GetBlock",
        "qldb:GetDigest",
        "qldb:GetRevision",
        "qldb:TagResource",
        "qldb:UntagResource",
        "qldb:ListTagsForResource",
        "qldb:SendCommand",
        "qldb:ExecuteStatement",
        "qldb:ShowCatalog",
        "qldb:InsertSampleData",
        "qldb:PartiQLCreateIndex",
        "qldb:PartiQLDropIndex",
        "qldb:PartiQLCreateTable",
        "qldb:PartiQLDropTable",
        "qldb:PartiQLUndropTable",
        "qldb:PartiQLDelete",
        "qldb:PartiQLInsert",
        "qldb:PartiQLUpdate",
        "qldb:PartiQLSelect",
        "qldb:PartiQLHistoryFunction"
```

```
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Action": [
        "kinesis:ListStreams",
        "kinesis:DescribeStream"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "qldb.amazonaws.com"
        }
      }
    }
  ]
}
```

Autorisation accordée aux utilisateurs pour afficher leurs propres autorisations

Cet exemple montre comment créer une politique qui permet aux utilisateurs IAM d'afficher les politiques en ligne et gérées attachées à leur identité d'utilisateur. Cette politique inclut les autorisations permettant d'effectuer cette action sur la console ou par programmation à l'aide de l'API AWS CLI or AWS .

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
```

```
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Transactions de données en cours

Pour interagir avec l'API de données transactionnelles QLDB (session QLDB) en exécutant des instructions partiQL sur un registre, vous devez autoriser l'action de l'API. `SendCommand` Le document JSON suivant est un exemple de politique qui autorise uniquement l'action de l'`SendCommand`API sur le registre`myExampleLedger`.

Pour utiliser cette politique, remplacez *`us-east-1`*, 123456789012*`, et`* dans l'exemple par vos propres informations. *`myExampleLedger`*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBSendCommandPermission",
            "Effect": "Allow",
            "Action": "qldb:SendCommand",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
        }
    ]
```

```
}
```

Si le mode `ALLOW_ALL` autorisations est `myExampleLedger` utilisé, cette politique accorde l'autorisation d'exécuter toutes les commandes partiQL sur n'importe quelle table du registre.

Vous pouvez également utiliser une politique AWS gérée pour accorder un accès complet à toutes les ressources QLDB. Pour plus d'informations, consultez AWS politiques gérées pour Amazon QLDB.

Autorisations standard pour les actions PartiQL et les ressources des tables

Pour les registres en mode `STANDARD` autorisations, vous pouvez vous référer aux documents de politique IAM suivants comme exemples d'octroi des autorisations partiQL appropriées. Pour obtenir la liste des autorisations requises pour chaque commande partiQL, consultez le. Référence des autorisations partiQL

Rubriques

- Accès complet à toutes les actions
- Accès complet à toutes les actions en fonction des balises du tableau
- Accès en lecture/écriture
- Accès en lecture seule
- Accès en lecture seule à une table spécifique
- Autoriser l'accès pour créer des tables
- Autoriser l'accès pour créer des tables en fonction des balises de demande

Accès complet à toutes les actions

Le document de politique JSON suivant accorde un accès complet pour utiliser toutes les commandes partiQL sur toutes les tables de. `myExampleLedger` Cette politique produit le même effet que l'utilisation du mode `ALLOW_ALL` autorisations pour le registre.

Pour utiliser cette politique, remplacez *`us-east-1`*, 123456789012*`, et`* dans l'exemple par vos propres informations. *`myExampleLedger`*

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "QLDBSendCommandPermission",
            "Effect": "Allow",
            "Action": "qldb:SendCommand",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
        },
        {
            "Sid": "QLDBPartiQLFullPermissions",
            "Effect": "Allow",
            "Action": [
                "qldb:PartiQLCreateIndex",
                "qldb:PartiQLDropIndex",
                "qldb:PartiQLCreateTable",
                "qldb:PartiQLDropTable",
                "qldb:PartiQLUndropTable",
                "qldb:PartiQLDelete",
                "qldb:PartiQLInsert",
                "qldb:PartiQLUpdate",
                "qldb:PartiQLRedact",
                "qldb:PartiQLSelect",
                "qldb:PartiQLHistoryFunction"
            ],
            "Resource": [
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/*",
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/
information_schema/user_tables"
            ]
        }
    ]
}
```

Accès complet à toutes les actions en fonction des balises du tableau

Le document de politique JSON suivant utilise une condition basée sur les balises de ressources de table pour accorder un accès complet à l'utilisation de toutes les commandes partiQL sur toutes les tables de. `myExampleLedger` Les autorisations ne sont accordées que si la balise de table `environment` possède la valeur`development`.

> **Warning**
>
> Il s'agit d'un exemple d'utilisation d'un caractère générique (*) pour autoriser toutes les actions PartiQL, y compris les opérations administratives et de lecture/écriture sur toutes

les tables d'un registre QLDB. Il est préférable de spécifier explicitement chaque action à accorder, et uniquement ce dont cet utilisateur, ce rôle ou ce groupe a besoin.

Pour utiliser cette politique, remplacez *us-east-1*, 123456789012*, et* dans l'exemple par vos propres informations. *myExampleLedger*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBSendCommandPermission",
            "Effect": "Allow",
            "Action": "qldb:SendCommand",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
        },
        {
            "Sid": "QLDBPartiQLFullPermissionsBasedOnTags",
            "Effect": "Allow",
            "Action": [
                "qldb:PartiQL*"
            ],
            "Resource": [
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/*",
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/
information_schema/user_tables"
            ],
            "Condition": {
                "StringEquals": { "aws:ResourceTag/environment": "development" }
            }
        }
    ]
}
```

Accès en lecture/écriture

Le document de politique JSON suivant autorise la sélection, l'insertion, la mise à jour et la suppression de données sur toutes les tables demyExampleLedger. Cette politique n'accorde pas l'autorisation de supprimer des données ou de modifier le schéma, par exemple pour créer et supprimer des tables et des index.

> **Note**
>
> Une UPDATE instruction nécessite des autorisations pour les `qldb:PartiQLSelect` actions `qldb:PartiQLUpdate` et sur la table en cours de modification. Lorsque vous exécutez une UPDATE instruction, celle-ci exécute une opération de lecture en plus de l'opération de mise à jour. Le fait d'exiger les deux actions garantit que seuls les utilisateurs autorisés à lire le contenu d'une table obtiennent UPDATE des autorisations.
> De même, une DELETE instruction nécessite des autorisations pour les `qldb:PartiQLSelect` actions `qldb:PartiQLDelete` et.

Pour utiliser cette politique, remplacez *us-east-1*, 123456789012*, et* dans l'exemple par vos propres informations. *myExampleLedger*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBSendCommandPermission",
            "Effect": "Allow",
            "Action": "qldb:SendCommand",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
        },
        {
            "Sid": "QLDBPartiQLReadWritePermissions",
            "Effect": "Allow",
            "Action": [
                "qldb:PartiQLDelete",
                "qldb:PartiQLInsert",
                "qldb:PartiQLUpdate",
                "qldb:PartiQLSelect",
                "qldb:PartiQLHistoryFunction"
            ],
            "Resource": [
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/*",
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/
information_schema/user_tables"
            ]
        }
    ]
}
```

Accès en lecture seule

Le document de politique JSON suivant accorde des autorisations de lecture seule sur toutes les tables de. `myExampleLedger` Pour utiliser cette politique, remplacez *`us-east-1`*, 123456789012*`,`* *`et`* dans l'exemple par vos propres informations. *`myExampleLedger`*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBSendCommandPermission",
            "Effect": "Allow",
            "Action": "qldb:SendCommand",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
        },
        {
            "Sid": "QLDBPartiQLReadOnlyPermissions",
            "Effect": "Allow",
            "Action": [
                "qldb:PartiQLSelect",
                "qldb:PartiQLHistoryFunction"
            ],
            "Resource": [
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/*",
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/
information_schema/user_tables"
            ]
        }
    ]
}
```

Accès en lecture seule à une table spécifique

Le document de politique JSON suivant accorde des autorisations de lecture seule sur une table spécifique dans. `myExampleLedger` Dans cet exemple, l'ID de table est`Au1EiThbt8s0z9wM26REZN`.

`Pour utiliser cette politique, remplacez us-east-1, 123456789012 et Au1 8S0z9WM26REZn dans l'exemple par vos myExampleLedgerpropres informations. EiThbt`

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBSendCommandPermission",
            "Effect": "Allow",
            "Action": "qldb:SendCommand",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
        },
        {
            "Sid": "QLDBPartiQLReadOnlyPermissionsOnTable",
            "Effect": "Allow",
            "Action": [
                "qldb:PartiQLSelect",
                "qldb:PartiQLHistoryFunction"
            ],
            "Resource": [
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/
table/Au1EiThbt8s0z9wM26REZN"
            ]
        }
    ]
}
```

Autoriser l'accès pour créer des tables

Le document de politique JSON suivant autorise la création de tables dans`myExampleLedger`. L'`qldb:PartiQLCreateTable`action nécessite des autorisations sur le type de ressource de table. Toutefois, l'ID d'une nouvelle table n'est pas connu au moment où vous exécutez une `CREATE TABLE` instruction. Ainsi, une politique qui accorde l'`qldb:PartiQLCreateTable`autorisation doit utiliser un caractère générique (*) dans l'ARN de la table pour spécifier la ressource.

Pour utiliser cette politique, remplacez *`us-east-1`*, 123456789012*`, et`* dans l'exemple par vos propres informations. *`myExampleLedger`*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBSendCommandPermission",
            "Effect": "Allow",
            "Action": "qldb:SendCommand",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
        },
```

```
        {
            "Sid": "QLDBPartiQLCreateTablePermission",
            "Effect": "Allow",
            "Action": [
                "qldb:PartiQLCreateTable"
            ],
            "Resource": [
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/*"
            ]
        }
    ]
}
```

Autoriser l'accès pour créer des tables en fonction des balises de demande

Le document de politique JSON suivant utilise une condition basée sur la clé de `aws:RequestTag` contexte pour accorder l'autorisation de créer des tables dans`myExampleLedger`. Les autorisations ne sont accordées que si la balise de demande `environment` possède la valeur`development`. Le balisage des tables lors de leur création nécessite l'accès aux `qldb:TagResource` actions `qldb:PartiQLCreateTable` et. Pour savoir comment étiqueter des tableaux lors de leur création, consultezTableaux de balisage.

Pour utiliser cette politique, remplacez *us-east-1*, 123456789012*, et* dans l'exemple par vos propres informations. *myExampleLedger*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBSendCommandPermission",
            "Effect": "Allow",
            "Action": "qldb:SendCommand",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
        },
        {
            "Sid": "QLDBPartiQLCreateTablePermission",
            "Effect": "Allow",
            "Action": [
                "qldb:PartiQLCreateTable",
                "qldb:TagResource"
            ],
            "Resource": [
```

```
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/*"
            ],
            "Condition": {
                "StringEquals": { "aws:RequestTag/environment": "development" }
            }
        }
    ]
}
```

Exportation d'un journal vers un compartiment Amazon S3

Étape 1 : Autorisations d'exportation du journal QLDB

Dans l'exemple suivant, vous accordez à un utilisateur l' Compte AWS autorisation d'effectuer l'`qldb:ExportJournalToS3`action sur une ressource de registre QLDB. Vous accordez également des autorisations pour effectuer l'`iam:PassRole`action sur la ressource de rôle IAM que vous souhaitez transmettre au service QLDB. Cela est obligatoire pour toutes les demandes d'exportation de journaux.

Pour utiliser cette politique, remplacez us-east-1, 123456789012 myExampleLedgeret qldb-s3-export dans l'exemple par vos propres informations.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBJournalExportPermission",
            "Effect": "Allow",
            "Action": "qldb:ExportJournalToS3",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
        },
        {
            "Sid": "IAMPassRolePermission",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::123456789012:role/qldb-s3-export",
            "Condition": {
               "StringEquals": {
                  "iam:PassedToService": "qldb.amazonaws.com"
               }
            }
```

```
        }
    ]
}
```

Étape 2 : autorisations du compartiment Amazon S3

Dans l'exemple suivant, vous utilisez un rôle IAM pour accorder à QLDB l'accès à l'écriture dans l'un de vos compartiments Amazon S3. `DOC-EXAMPLE-BUCKET` Cela est également requis pour toutes les exportations de journaux QLDB.

Outre l'octroi de l'`s3:PutObject`autorisation, la politique accorde également l'`s3:PutObjectAcl`autorisation de définir les autorisations de la liste de contrôle d'accès (ACL) pour un objet.

Pour utiliser cette politique, remplacez DOC-EXAMPLE-BUCKET dans l'exemple par le nom de votre compartiment Amazon S3.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBJournalExportS3Permissions",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        }
    ]
}
```

Vous associez ensuite cette politique d'autorisation à un rôle IAM que QLDB peut assumer pour accéder à votre compartiment Amazon S3. Le document JSON suivant est un exemple de politique de confiance qui permet à QLDB d'assumer le rôle IAM pour n'importe quelle ressource QLDB du compte uniquement. `123456789012`

Pour utiliser cette politique, remplacez *`us-east-1` et* 123456789012 dans l'exemple par vos propres informations.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "qldb.amazonaws.com"
            },
            "Action": [ "sts:AssumeRole" ],
            "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:qldb:us-east-1:123456789012:*"
                },
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                }
            }
        }
    ]
}
```

Diffusion d'un journal sur Kinesis Data Streams

Étape 1 : autorisations du flux de journaux QLDB

Dans l'exemple suivant, vous accordez à un utilisateur l' Compte AWS autorisation d'effectuer l'`qldb:StreamJournalToKinesis`action sur toutes les sous-ressources de flux QLDB d'un registre. Vous accordez également des autorisations pour effectuer l'`iam:PassRole`action sur la ressource de rôle IAM que vous souhaitez transmettre au service QLDB. Cela est obligatoire pour toutes les demandes de flux de journal.

Pour utiliser cette politique, remplacez *us-east-1*, 123456789012 *myExampleLedger*et, dans l'exemple, par vos propres informations. *qldb-kinesis-stream*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBJournalStreamPermission",
            "Effect": "Allow",
            "Action": "qldb:StreamJournalToKinesis",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:stream/myExampleLedger/*"
        },
        {
```

```
            "Sid": "IAMPassRolePermission",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::123456789012:role/qldb-kinesis-stream",
            "Condition": {
               "StringEquals": {
                  "iam:PassedToService": "qldb.amazonaws.com"
               }
            }
        }
    ]
}
```

Étape 2 : autorisations Kinesis Data Streams

Dans l'exemple suivant, vous utilisez un rôle IAM pour autoriser QLDB à écrire des enregistrements de données dans votre flux de données Amazon Kinesis. *stream-for-qldb* Cela est également requis pour tous les flux de journaux QLDB.

Pour utiliser cette politique, remplacez *us-east-1*, 123456789012*, et* dans l'exemple par vos propres informations. *stream-for-qldb*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBStreamKinesisPermissions",
            "Action": [ "kinesis:PutRecord*", "kinesis:DescribeStream",
 "kinesis:ListShards" ],
            "Effect": "Allow",
            "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/stream-for-qldb"
        }
    ]
}
```

Vous associez ensuite cette politique d'autorisation à un rôle IAM que QLDB peut assumer pour accéder à votre flux de données Kinesis. Le document JSON suivant est un exemple de politique de confiance qui permet à QLDB d'assumer un rôle IAM pour n'importe quel flux QLDB dans le compte du registre uniquement. `123456789012` `myExampleLedger`

Pour utiliser cette politique, remplacez *us-east-1*, 123456789012*, et* dans l'exemple par vos propres informations. *myExampleLedger*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "qldb.amazonaws.com"
            },
            "Action": [ "sts:AssumeRole" ],
            "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:qldb:us-
east-1:123456789012:stream/myExampleLedger/*"
                },
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                }
            }
        }
    ]
}
```

Mise à jour des registres QLDB en fonction des balises

Vous pouvez utiliser des conditions dans votre politique basée sur l'identité pour contrôler l'accès aux ressources QLDB en fonction de balises. Cet exemple montre comment créer une politique permettant de mettre à jour un registre. Toutefois, l'autorisation n'est accordée que si le tag Ledger `Owner` a la valeur du nom d'utilisateur de cet utilisateur. Cette politique accorde également les autorisations nécessaires pour réaliser cette action sur la console.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListLedgersInConsole",
            "Effect": "Allow",
            "Action": "qldb:ListLedgers",
            "Resource": "*"
        },
        {
            "Sid": "UpdateLedgerIfOwner",
            "Effect": "Allow",
```

```
            "Action": "qldb:UpdateLedger",
            "Resource": "arn:aws:qldb:*:*:ledger/*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/Owner": "${aws:username}"}
            }
        }
    ]
}
```

Vous pouvez attacher cette stratégie aux utilisateurs de votre compte. Si un utilisateur nommé `richard-roe` essaie de mettre à jour un registre QLDB, le registre doit être étiqueté ou. `Owner=richard-roe` `owner=richard-roe` Dans le cas contraire, l'utilisateur se voit refuser l'accès. La clé de condition de balise `Owner` correspond à la fois à `Owner` et à `owner`, car les noms de clé de condition ne sont pas sensibles à la casse. Pour plus d'informations, consultez Conditions pour éléments de politique JSON IAM dans le Guide de l'utilisateur IAM.

Prévention du cas de figure de l'adjoint désorienté entre services

Le problème de député confus est un problème de sécurité dans lequel une entité qui n'est pas autorisée à effectuer une action peut contraindre une entité plus privilégiée à le faire. En AWS, l'usurpation d'identité interservices peut entraîner la confusion des adjoints.

L'usurpation d'identité entre services peut se produire lorsqu'un service (le service appelant) appelle un autre service (le service appelé). Le service appelant peut être manipulé et ses autorisations utilisées pour agir sur les ressources d'un autre client auxquelles on ne serait pas autorisé d'accéder autrement. Pour éviter le problème de confusion des adjoints, AWS fournit des outils qui vous aident à protéger vos données pour tous les services, les responsables de service ayant accès aux ressources de votre compte.

Nous recommandons d'utiliser les clés de contexte de condition `aws:SourceAccount`globale `aws:SourceArn`et les clés de contexte dans les politiques de ressources afin de limiter les autorisations qu'Amazon QLDB accorde à un autre service à la ressource. Si vous utilisez les deux clés contextuelles de condition globale, la `aws:SourceAccount` valeur et le compte figurant dans la `aws:SourceArn` valeur doivent utiliser le même identifiant de compte lorsqu'ils sont utilisés dans la même déclaration de politique.

Le tableau suivant répertorie les valeurs possibles de `aws:SourceArn` pour les opérations d'API `StreamsJournalToKinesis`QLDB `ExportJournalToS3`et de QLDB. Ces opérations sont

concernées par ce problème de sécurité car elles appellent AWS Security Token Service (AWS STS) pour assumer le rôle IAM que vous spécifiez.

Opération API	Service appelé	lois : SourceArn
`ExportJournalToS3`	AWS STS (AssumeRole)	Permet à QLDB d'assumer le rôle de toutes les ressources QLDB du compte : `arn:aws:qldb: us-east-1 :123456789012 :*` Actuellement, QLDB ne prend en charge cet ARN générique que pour les exportations de journaux.

`StreamsJournalToKinesis`	AWS STS (AssumeRole)	Permet à QLDB d'assumer le rôle d'un flux QLDB spécifique :
		`arn:aws:qldb: us-east-1 :123456789012 :stream/myExample Ledger /IiPT4brpZCqCq3f4MTHbYy`
		Remarque : Vous ne pouvez spécifier un ID de flux dans l'ARN qu'une fois la ressource de flux créée. À l'aide de cet ARN, vous pouvez autoriser l'utilisation du rôle uniquement pour un seul flux QLDB.
		Permet à QLDB d'assumer le rôle de tous les flux QLDB d'un registre :
		`arn:aws:qldb: us-east-1 :123456789012 :stream/myExample Ledger /*`
		Permet à QLDB d'assumer le rôle de tous les flux QLDB du compte :
		`arn:aws:qldb: us-east-1 :123456789012 :stream/*`
		Permet à QLDB d'assumer le rôle de toutes les ressources QLDB du compte :

Opération API	Service appelé	lois : SourceArn
		`arn:aws:qldb: us-east-1 :123456789012 :*`

Le moyen le plus efficace de se protéger contre le problème de député confus consiste à utiliser la clé de contexte de condition globale `aws:SourceArn` avec l'ARN complet de la ressource. Si vous ne connaissez pas l'ARN complet de la ressource ou si vous spécifiez plusieurs ressources, utilisez la clé de condition contextuelle `aws:SourceArn` globale avec des caractères génériques (*) pour les parties inconnues de l'ARN, par exemple,. `arn:aws:qldb:us-east-1:123456789012:*`

L'exemple de politique de confiance suivant pour un rôle IAM montre comment vous pouvez utiliser les clés contextuelles `aws:SourceArn` et de condition `aws:SourceAccount` globale pour éviter le problème de confusion des adjoints. Grâce à cette politique de confiance, QLDB peut assumer le rôle de n'importe quel flux QLDB dans le compte uniquement pour le registre. `123456789012` `myExampleLedger`

Pour utiliser cette politique, remplacez *us-east-1*, 123456789012*, et* dans l'exemple par vos propres informations. *myExampleLedger*

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "qldb.amazonaws.com"
    },
    "Action": [ "sts:AssumeRole" ],
    "Condition": {
      "ArnEquals": {
        "aws:SourceArn": "arn:aws:qldb:us-east-1:123456789012:stream/myExampleLedger/*"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

AWS politiques gérées pour Amazon QLDB

Une politique AWS gérée est une politique autonome créée et administrée par AWS. AWS les politiques gérées sont conçues pour fournir des autorisations pour de nombreux cas d'utilisation courants afin que vous puissiez commencer à attribuer des autorisations aux utilisateurs, aux groupes et aux rôles.

N'oubliez pas que les politiques AWS gérées peuvent ne pas accorder d'autorisations de moindre privilège pour vos cas d'utilisation spécifiques, car elles sont accessibles à tous les AWS clients. Nous vous recommandons de réduire encore les autorisations en définissant des politiques gérées par le client qui sont propres à vos cas d'utilisation.

Vous ne pouvez pas modifier les autorisations définies dans les politiques AWS gérées. Si les autorisations définies dans une politique AWS gérée sont AWS mises à jour, la mise à jour affecte toutes les identités principales (utilisateurs, groupes et rôles) auxquelles la politique est attachée. AWS est le plus susceptible de mettre à jour une politique AWS gérée lorsqu'une nouvelle politique Service AWS est lancée ou lorsque de nouvelles opérations d'API sont disponibles pour les services existants.

Pour plus d'informations, consultez la section Politiques gérées par AWS dans le Guide de l'utilisateur IAM.

Pour plus d'informations sur les opérations de l'API QLDB dans AWS ces politiques gérées, consultez le. Référence d'API Amazon QLDB

Rubriques

- AWS politique gérée : AmazonQLDB ReadOnly
- AWS politique gérée : AmazonQLDB FullAccess
- AWS politique gérée : AmazonQLDB ConsoleFullAccess
- Mises à jour des politiques gérées par QLDB AWS

AWS politique gérée : AmazonQLDB ReadOnly

Utilisez la ReadOnly politique d'AmazonQLDB pour accorder des autorisations de lecture seule à toutes les ressources QLDB. Vous pouvez associer cette politique à vos identités IAM.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes pour le `qldb` service.

- Permet aux principaux de décrire et de répertorier toutes les ressources QLDB et leurs balises. Ces ressources incluent les registres, les tâches d'exportation Amazon S3 et les flux vers Kinesis Data Streams.
- Permet aux directeurs d'obtenir un bloc, un résumé ou une révision du journal dans n'importe quel registre afin de vérifier les données de manière cryptographique.
- N'autorise pas les principaux à exécuter des commandes partiQL sur aucune table dans aucun registre.

AWS politique gérée : AmazonQLDB FullAccess

Utilisez la FullAccess politique d'AmazonQLDB pour accorder des autorisations administratives complètes à toutes les ressources QLDB via l'API QLDB ou le. AWS CLI Vous pouvez associer cette politique à vos identités IAM.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes.

- `qldb`
  - Permet aux principaux de créer, de décrire, de répertorier et de gérer toutes les ressources QLDB et leurs balises. Ces ressources incluent les registres, les tâches d'exportation Amazon S3 et les flux vers Kinesis Data Streams.
  - Permet aux principaux d'exécuter toutes les commandes partiQL sur toutes les tables de n'importe quel registre à l'aide du pilote QLDB ou du shell QLDB.
  - Permet aux directeurs d'obtenir un bloc, un résumé ou une révision du journal dans n'importe quel registre afin de vérifier les données de manière cryptographique.
- `iam`— Permet aux principaux de transmettre n'importe quelle ressource de rôle IAM de votre compte au service QLDB. Cela est obligatoire pour toutes les demandes d'exportation et de diffusion de journaux.

AWS politique gérée : AmazonQLDB ConsoleFullAccess

Utilisez la ConsoleFullAccess politique d'AmazonQLDB pour accorder des autorisations administratives complètes à toutes les ressources QLDB via l'API QLDB ou AWS Management Console le. AWS CLI Vous pouvez associer cette politique à vos identités IAM.

Détails de l’autorisation

Cette politique inclut les autorisations suivantes.

- `qldb`
  - Permet aux principaux de créer, de décrire, de répertorier et de gérer toutes les ressources QLDB et leurs balises. Ces ressources incluent les registres, les tâches d'exportation Amazon S3 et les flux vers Kinesis Data Streams.
  - Permet aux principaux d'exécuter toutes les commandes partiQL sur toutes les tables de n'importe quel registre à l'aide de la console QLDB, du pilote QLDB ou du shell QLDB.
  - Permet aux principaux d'insérer des exemples de données d'application dans n'importe quel registre à l'aide de la console QLDB.
  - Permet aux directeurs d'obtenir un bloc, un résumé ou une révision du journal dans n'importe quel registre afin de vérifier les données de manière cryptographique.
- `dbqms`— Permet aux principaux d'utiliser toutes les actions du service de métadonnées de requête de base de données. Il s'agit d'un service interne uniquement dont la console QLDB a besoin pour créer, décrire et gérer les requêtes récentes et enregistrées pour l'éditeur de requêtes partiQL.
- `kinesis`— Permet aux directeurs de décrire et de répertorier les ressources Amazon Kinesis Data Streams. Ces ressources sont les destinations cibles vers lesquelles les ressources de flux QLDB peuvent écrire des données.
- `iam`— Permet aux principaux de transmettre n'importe quelle ressource de rôle IAM de votre compte au service QLDB. Cela est obligatoire pour toutes les demandes d'exportation et de diffusion de journaux.

Mises à jour des politiques gérées par QLDB AWS

Consultez les détails des mises à jour apportées aux politiques AWS gérées pour QLDB depuis que ce service a commencé à suivre ces modifications. Pour recevoir des alertes automatiques

concernant les modifications apportées à cette page, abonnez-vous au flux RSS sur la page d'historique des versions de QLDB.

Modification	Description	Date
AmazonQLDBFullAccess, AmazonQLDB — Mise à jour des politiques ConsoleFullAccess existantes	QLDB a ajouté une nouvelle autorisation pour permettre aux principaux de rédiger les révisions de documents dans tous les registres en mode autorisations. `STANDARD`	4 novembre 2022
AmazonQLDBFullAccess, AmazonQLDB — Mise à jour des politiques ConsoleFullAccess existantes	QLDB a ajouté de nouvelles autorisations pour permettre aux principaux de transmettre n'importe quelle ressource de rôle IAM de votre compte au service QLDB. Cela est obligatoire pour toutes les demandes d'exportation et de diffusion de journaux.	2 septembre 2021
AmazonQLDB ReadOnly — Mise à jour d'une politique existante	QLDB a supprimé une action `qldb:GetBlock` dupliquée précédemment répertoriée deux fois et a réorganisé `"Effect"` le champ afin qu'il apparaisse avant le champ. `"Action"`	1er juillet 2021
AmazonQLDBFullAccess, AmazonQLDB — Mise à jour des politiques ConsoleFullAccess existantes	QLDB a ajouté de nouvelles autorisations pour permettre aux principaux de mettre à jour le mode d'autorisations dans tous les registres et d'exécuter toutes les commandes partiQL dans	27 mai 2021

Modification	Description	Date
	tous les registres dans le nouveau mode d'autorisations. `STANDARD` Le mode `STANDARD` autorisations prend en charge le contrôle d'accès au niveau de la table et la granularité pour les commandes partiQL. Pour faciliter le nouveau mode d'autorisations, QLDB a introduit un ensemble d'actions IAM pour les types de commandes partiQL et Amazon Resource Names (ARN) pour les ressources des tables QLDB. Ces deux politiques sont mises à jour pour inclure les nouvelles actions partiQL permettant d'accorder un accès complet aux `STANDARD` registres.	
QLDB a commencé à suivre les modifications	QLDB a commencé à suivre les modifications apportées à ses politiques gérées. AWS	1er mars 2021

Résolution des problèmes d'identité et d'accès à Amazon QLDB

Utilisez les informations suivantes pour vous aider à diagnostiquer et à résoudre les problèmes courants que vous pouvez rencontrer lorsque vous travaillez avec QLDB et IAM.

Rubriques

- Je ne suis pas autorisé à effectuer une action dans QLDB
- Je ne suis pas autorisé à effectuer iam : PassRole

- Je souhaite autoriser des personnes extérieures à moi Compte AWS à accéder à mes ressources QLDB

Je ne suis pas autorisé à effectuer une action dans QLDB

S'il vous AWS Management Console indique que vous n'êtes pas autorisé à effectuer une action, vous devez contacter votre administrateur pour obtenir de l'aide. Votre administrateur est la personne qui vous a fourni vos informations de connexion.

L'exemple d'erreur suivant se produit quand l'utilisateur `mateojackson` tente d'utiliser la console pour afficher des informations détaillées sur une ressource *`myExampleLedger`* fictive, mais ne dispose pas des autorisations `qldb:`*`DescribeLedger`* fictives.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 qldb:DescribeLedger on resource: myExampleLedger
```

Dans ce cas, Mateo demande à son administrateur de mettre à jour ses politiques pour lui permettre d'accéder à la ressource *`myExampleLedger`* à l'aide de l'action `qldb:`*`DescribeLedger`*.

Je ne suis pas autorisé à effectuer iam : PassRole

Si vous recevez un message d'erreur indiquant que vous n'êtes pas autorisé à effectuer l'`iam:PassRole`action, vos politiques doivent être mises à jour pour vous permettre de transmettre un rôle à QLDB.

Certains vous Services AWS permettent de transmettre un rôle existant à ce service au lieu de créer un nouveau rôle de service ou un rôle lié à un service. Pour ce faire, un utilisateur doit disposer des autorisations nécessaires pour transmettre le rôle au service.

L'exemple d'erreur suivant se produit lorsqu'un utilisateur IAM nommé `marymajor` essaie d'utiliser la console pour effectuer une action dans QLDB. Toutefois, l'action nécessite que le service ait des autorisations accordées par un rôle de service. Mary ne dispose pas des autorisations nécessaires pour transférer le rôle au service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dans ce cas, les politiques de Mary doivent être mises à jour pour lui permettre d'exécuter l'action `iam:PassRole`.

Si vous avez besoin d'aide, contactez votre AWS administrateur. Votre administrateur vous a fourni vos informations d'identification de connexion.

Pour obtenir des conseils de résolution de cette erreur spécifiques aux opérations d'exportation ou de diffusion de journaux, consultezRésolutions des problèmes liés à Amazon QLDB.

Je souhaite autoriser des personnes extérieures à moi Compte AWS à accéder à mes ressources QLDB

Vous pouvez créer un rôle que les utilisateurs provenant d'autres comptes ou les personnes extérieures à votre organisation pourront utiliser pour accéder à vos ressources. Vous pouvez spécifier qui est autorisé à assumer le rôle. Pour les services qui prennent en charge les politiques basées sur les ressources ou les listes de contrôle d'accès (ACL), vous pouvez utiliser ces politiques pour donner l'accès à vos ressources.

Pour en savoir plus, consultez les éléments suivants :

- Pour savoir si QLDB prend en charge ces fonctionnalités, consultez. Comment Amazon QLDB fonctionne avec IAM
- Pour savoir comment fournir l'accès à vos ressources sur celles Comptes AWS que vous possédez, consultez la section Fournir l'accès à un utilisateur IAM dans un autre utilisateur Compte AWS que vous possédez dans le Guide de l'utilisateur IAM.
- Pour savoir comment fournir l'accès à vos ressources à des tiers Comptes AWS, consultez la section Fournir un accès à des ressources Comptes AWS détenues par des tiers dans le guide de l'utilisateur IAM.
- Pour savoir comment fournir un accès par le biais de la fédération d'identité, consultez Fournir un accès à des utilisateurs authentifiés en externe (fédération d'identité) dans le Guide de l'utilisateur IAM.
- Pour découvrir quelle est la différence entre l'utilisation des rôles et l'utilisation des politiques basées sur les ressources pour l'accès entre comptes, consultez Différence entre les rôles IAM et les politiques basées sur les ressources dans le Guide de l'utilisateur IAM.

Journalisation et surveillance dans Amazon QLDB

La surveillance joue un rôle important dans le maintien de la fiabilité, de la disponibilité et des performances d'Amazon QLDB et de vos solutions. AWS Vous devez collecter des données de surveillance provenant de toutes les parties de votre AWS solution afin de pouvoir corriger plus

facilement une défaillance multipoint, le cas échéant. Toutefois, avant de commencer à surveiller QLDB, vous devez créer un plan de surveillance comprenant des réponses aux questions suivantes :

- Quels sont les objectifs de la surveillance ?
- Quelles sont les ressources à surveiller ?
- À quelle fréquence les ressources doivent-elles être surveillées ?
- Quels outils de surveillance utiliser ?
- Qui exécute les tâches de supervision ?
- Qui doit être informé en cas de problème ?

L'étape suivante consiste à établir une base de référence pour les performances QLDB normales dans votre environnement, en mesurant les performances à différents moments et dans différentes conditions de charge. Lorsque vous surveillez QLDB, stockez les données de surveillance historiques afin de pouvoir les comparer aux données de performance actuelles, d'identifier les modèles de performances normaux et les anomalies de performances, et de concevoir des méthodes pour résoudre les problèmes.

Pour établir une référence, vous devez, au moins, superviser les éléments suivants :

- Lecture et écriture des E/S et du stockage, afin de pouvoir suivre les habitudes de consommation de votre registre à des fins de facturation.
- Latence des commandes, afin que vous puissiez suivre les performances de votre registre lors de l'exécution d'opérations de données.
- Exceptions, afin que vous puissiez déterminer si des demandes ont entraîné une erreur.

Rubriques

Outils de surveillance

AWS fournit différents outils que vous pouvez utiliser pour surveiller Amazon QLDB. Vous pouvez configurer certains outils pour qu'ils effectuent la supervision automatiquement, tandis que d'autres

nécessitent une intervention manuelle. Nous vous recommandons d'automatiser le plus possible les tâches de supervision.

Rubriques

Outils de surveillance automatique

Vous pouvez utiliser les outils de surveillance automatique suivants pour surveiller QLDB et signaler tout problème :

- Amazon CloudWatch Alarms — Surveillez une seule métrique sur une période que vous spécifiez et effectuez une ou plusieurs actions en fonction de la valeur de la métrique par rapport à un seuil donné sur un certain nombre de périodes. L'action est une notification envoyée à une rubrique Amazon Simple Notification Service (Amazon SNS) ou à une politique Amazon EC2 Auto Scaling. CloudWatch les alarmes n'appellent pas d'actions simplement parce qu'elles sont dans un état particulier ; l'état doit avoir changé et être maintenu pendant un certain nombre de périodes. Pour plus d’informations, consultez Surveillance avec Amazon CloudWatch.
- Amazon CloudWatch Logs — Surveillez, stockez et accédez à vos fichiers journaux depuis AWS CloudTrail ou d'autres sources. Pour plus d'informations, consultez la section Surveillance des fichiers journaux dans le guide de CloudWatch l'utilisateur Amazon.
- Amazon CloudWatch Events : associez les événements et acheminez-les vers une ou plusieurs fonctions ou flux cibles afin d'apporter des modifications, de recueillir des informations d'état et de prendre des mesures correctives. Pour plus d'informations, consultez la section Qu'est-ce qu'Amazon CloudWatch Events dans le guide de CloudWatch l'utilisateur Amazon.
- AWS CloudTrail Surveillance des journaux : partagez les fichiers journaux entre les comptes, surveillez les fichiers CloudTrail CloudWatch journaux en temps réel en les envoyant à Logs, écrivez des applications de traitement des journaux en Java et vérifiez que vos fichiers journaux n'ont pas changé après leur livraison par CloudTrail. Pour plus d'informations, consultez la section Utilisation des fichiers CloudTrail journaux dans le Guide de AWS CloudTrail l'utilisateur.

Outils de surveillance manuelle

Un autre élément important de la surveillance de QLDB consiste à surveiller manuellement les éléments non couverts par CloudWatch les alarmes. Les tableaux de bord QLDB CloudWatch,,

Trusted Advisor, et AWS Management Console autres fournissent at-a-glance une vue de l'état de votre environnement. AWS Nous vous recommandons de consulter également les fichiers journaux sur Amazon QLDB.

- Le tableau de bord QLDB affiche les éléments suivants :
  - E/S en lecture et en écriture
  - Journal et stockage indexé
  - Latence des commandes
  - Exceptions
- La page CloudWatch d'accueil affiche les informations suivantes :
  - Alarmes et statuts en cours
  - Graphiques des alarmes et des ressources
  - Statut d’intégrité du service

  En outre, vous pouvez CloudWatch effectuer les opérations suivantes :
  - Créer des tableaux de bord personnalisés pour surveiller les services de votre choix
  - Représenter graphiquement les données de métriques pour résoudre les problèmes et découvrir les tendances
  - Recherchez et parcourez tous les indicateurs de vos AWS ressources
  - Créer et modifier des alarmes pour être informé des problèmes

Surveillance avec Amazon CloudWatch

Vous pouvez surveiller Amazon QLDB à CloudWatch l'aide de métriques lisibles, qui collecte et traite les données brutes d'Amazon QLDB. near-real-time Ces statistiques sont enregistrées pour une durée de deux semaines ; par conséquent, vous pouvez accéder aux informations historiques et acquérir un meilleur point de vue de la façon dont votre service ou application web s'exécute. Par défaut, les données métriques QLDB sont automatiquement envoyées dans des périodes de 1 CloudWatch ou 15 minutes. Pour plus d'informations, consultez Que sont Amazon CloudWatch, Amazon CloudWatch Events et Amazon CloudWatch Logs ? dans le guide de CloudWatch l'utilisateur Amazon.

Rubriques

- Comment utiliser les métriques QLDB ?

- Mesures et dimensions d'Amazon QLDB
- Création d' CloudWatch alarmes pour surveiller Amazon QLDB

Comment utiliser les métriques QLDB ?

Les métriques rapportées par QLDB fournissent des informations que vous pouvez analyser de différentes manières. La liste suivante présente certaines utilisations courantes des métriques. Voici quelques suggestions pour vous aider à démarrer, qui ne forment pas une liste exhaustive.

- Vous pouvez surveiller `JournalStorage` et `IndexedStorage` sur une période donnée, pour suivre la quantité d'espace disque consommée par votre registre.
- Vous pouvez surveiller `ReadIOs` et `WriteIOs` sur une période donnée le nombre de demandes traitées par votre registre.
- Vous pouvez surveiller `CommandLatency` pour suivre les performances de votre registre en matière d'opérations de données et analyser les types de commandes qui génèrent le plus de latence.

Mesures et dimensions d'Amazon QLDB

Lorsque vous interagissez avec Amazon QLDB, celui-ci envoie les métriques et dimensions suivantes à. CloudWatch Les métriques de stockage sont rapportées toutes les 15 minutes, et toutes les autres métriques sont agrégées et rapportées toutes les minutes. Vous pouvez utiliser les procédures suivantes pour afficher les métriques de QLDB.

Pour afficher les métriques à l'aide de la CloudWatch console

Les métriques sont d'abord regroupées par espace de noms de service, puis par les différentes combinaisons de dimension au sein de chaque espace de noms.

1. Ouvrez la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Si nécessaire, changez la région. Dans la barre de navigation, choisissez la région dans laquelle se trouvent vos AWS ressources. Pour de plus amples informations, veuillez consulter Régions et points de terminaison.
3. Dans le panneau de navigation, sélectionnez Metrics (Métriques).
4. Dans l'onglet Toutes les mesures, choisissez QLDB.

Pour consulter les statistiques à l'aide du AWS CLI

- À partir d'une invite de commande, utilisez la commande suivante :

```
aws cloudwatch list-metrics --namespace "AWS/QLDB"
```

CloudWatch affiche les mesures suivantes pour QLDB.

Dimensions et statistiques d'Amazon QLDB

Les métriques et les dimensions qu'Amazon QLDB envoie à CloudWatch Amazon sont répertoriées ici.

Métriques QLDB

Métrique	Description
`JournalStorage`	La quantité totale d'espace disque utilisée par le journal du registre, indiquée à intervalles de 15 minutes. Le journal contient l'historique complet, immuable et vérifiabl e de toutes les modifications apportées à vos données. Unités : `Bytes` Dimensions : `LedgerName`
`IndexedStorage`	Quantité totale d'espace disque utilisée par les tables, les index et l'historique indexé du registre, indiquée à intervall es de 15 minutes. Le stockage indexé est constitué de données de registre optimisées pour les requêtes à hautes performances. Unités : `Bytes` Dimensions : `LedgerName`
`ReadIOs`	Le nombre de demandes d'E/S de lecture, indiqué à intervalles d'une minute. Cela capture tous les types d'opérations de lecture, y compris les transactions de

Métrique	Description
	données, les demandes de vérification, les exportations de journaux et les flux de journaux. Unités : `Count` Dimensions : `LedgerName`
`WriteIOs`	Le nombre de demandes d'E/S d'écriture, indiqué à intervalles d'une minute. Unités : `Count` Dimensions : `LedgerName`
`CommandLatency`	Le temps nécessaire aux opérations sur les données, indiqué à intervalles d'une minute. Unités : `Milliseconds` Dimensions : `CommandType, LedgerName`
`IsImpaired`	L'indicateur qui indique si un flux de journal vers Kinesis Data Streams est altéré, signalé à intervalles d'une minute. La valeur de `1` indique que le flux est dans un état altéré, et `0` indique le contraire. Unités : `Boolean` (0 ou 1) Dimensions : `LedgerName, StreamId`
`OccConflictExceptions`	Le nombre de demandes adressées à QLDB qui génèrent un. `OccConflictException` Pour plus d'informations sur le contrôle de simultanéité optimiste (OCC), consultez. Modèle de mise QLDB simultansimultansimultansimultansimultansimultansimultansimultansimultansimultansimultansimultansimultansimultanéité Unités `Count`:

Métrique	Description
`Session4xxExceptions`	Nombre de requêtes adressées à QLDB qui génèrent une erreur HTTP 4xx. Unités `Count`:
`Session5xxExceptions`	Nombre de requêtes adressées à QLDB qui génèrent une erreur HTTP 5xx. Unités `Count`:
`SessionRateExceededExceptions`	Le nombre de demandes adressées à QLDB qui génèrent un. `SessionRateExceededException` Unités `Count`:

Dimensions pour les métriques QLDB

Les métriques de QLDB sont qualifiées par les valeurs du compte, du nom du registre, de l'ID de flux ou du type de commande. Vous pouvez utiliser la CloudWatch console pour récupérer des données QLDB selon l'une des dimensions du tableau suivant.

Dimension	Description
`LedgerName`	Cette dimension limite les données à un registre spécifique. Cette valeur peut être n'importe quel nom de registre en cours ou Région AWS en cours Compte AWS.
`StreamId`	Cette dimension limite les données à un flux de journal spécifique. Cette valeur peut être n'importe quel identifiant de flux pour un registre en cours Région AWS et en cours Compte AWS.
`CommandType`	Cette dimension limite les données à l'une des commandes de l'API de données QLDB suivantes : • `AbortTransaction` • `CommitTransaction`

Dimension	Description
	• `EndSession` • `ExecuteStatement` • `FetchPage` • `StartSession` • `StartTransaction` Pour savoir comment QLDB utilise ces commandes pour gérer les opérations de données, consultez. Gestion des sessions avec le chauffeur

Création d' CloudWatch alarmes pour surveiller Amazon QLDB

Vous pouvez créer une CloudWatch alarme Amazon qui envoie un message Amazon Simple Notification Service (Amazon SNS) lorsque l'état de l'alarme change. Une alarme surveille une seule métrique pendant la période que vous spécifiez. Elle réalise une ou plusieurs actions en fonction de la valeur de la métrique par rapport à un seuil donné sur un certain nombre de périodes. L'action est une notification envoyée à une rubrique Amazon SNS ou à une politique Auto Scaling.

Les alarmes déclenchent des actions uniquement pour les changements d'état prolongés. CloudWatch les alarmes n'appellent pas d'actions simplement parce qu'elles sont dans un état particulier. L'état doit avoir changé et avoir été maintenu pendant un nombre de périodes spécifié.

Pour plus d'informations sur la création d' CloudWatch alarmes, consultez la section Utilisation des CloudWatch alarmes Amazon dans le guide de CloudWatch l'utilisateur Amazon.

Automatisation d'Amazon QLDB avec des événements CloudWatch

Amazon CloudWatch Events vous permet d'automatiser Services AWS et de répondre automatiquement aux événements du système tels que les problèmes de disponibilité des applications ou les modifications des ressources. Les événements de Services AWS sont transmis à CloudWatch Events en temps quasi réel. Vous pouvez écrire des règles simples pour indiquer quels événements vous intéressent et les actions automatisées à effectuer quand un événement correspond à une règle. Les actions pouvant être déclenchées automatiquement sont les suivantes :

- Invoquer une fonction AWS Lambda

- Appel de la fonctionnalité Exécuter la commande d'Amazon EC2
- Relais de l'événement à Amazon Kinesis Data Streams
- Activation d'une machine à AWS Step Functions états
- Notification d'une rubrique Amazon SNS ou d'une file d'attente Amazon SQS

Amazon QLDB signale un événement CloudWatch à Events chaque fois que l'état d'une ressource de registre change dans vos fichiers. Compte AWS Les événements sont actuellement émis sur une at-least-once base garantie pour les ressources du registre QLDB uniquement.

Voici un exemple d'événement signalé par QLDB, au cours duquel l'état d'un registre est passé à. `DELETING`

```
{
    "version" : "0",
    "id" : "2f6557eb-e361-54ef-0f9f-99dd9f171c62",
    "detail-type" : "QLDB Ledger State Change",
    "source" : "aws.qldb",
    "account" : "123456789012",
    "time" : "2019-07-24T21:59:17Z",
    "region" : "us-east-1",
    "resources" : ["arn:aws:qldb:us-east-1:123456789012:ledger/exampleLedger"],
    "detail" : {
        "ledgerName" : "exampleLedger",
        "state" : "DELETING"
    }
}
```

Voici quelques exemples d'utilisation d' CloudWatch Events avec QLDB :

- Activation d'une fonction Lambda chaque fois qu'un nouveau registre est initialement créé dans son `CREATING` état et le devient finalement. `ACTIVE`
- Notifier un sujet Amazon SNS lorsque l'état de votre registre change de puis `DELETING` de. `DELETED`

Pour plus d'informations, consultez le guide de l'utilisateur d'Amazon CloudWatch Events.

Journalisation des appels d'API Amazon QLDB avec AWS CloudTrail

Amazon QLDB est intégré AWS CloudTrailà un service qui fournit un enregistrement des actions entreprises par un utilisateur, un rôle ou un dans QLDB. Service AWS CloudTrail capture tous les appels d'API de gestion des ressources pour QLDB sous forme d'événements. Les appels capturés incluent les appels provenant de la console QLDB et les appels de code vers les opérations de l'API QLDB. Si vous créez un suivi, vous pouvez activer la diffusion continue des CloudTrail événements vers un bucket Amazon Simple Storage Service (Amazon S3), y compris les événements pour QLDB. Si vous ne configurez pas de suivi, vous pouvez toujours consulter les événements les plus récents sur la CloudTrail console dans Historique des événements. À l'aide des informations collectées par CloudTrail, vous pouvez déterminer la demande qui a été faite à QLDB, l'adresse IP à partir de laquelle la demande a été faite, l'auteur de la demande, la date à laquelle elle a été faite et des informations supplémentaires.

Pour en savoir plus CloudTrail, notamment comment le configurer et l'activer, consultez le guide de AWS CloudTrail l'utilisateur.

Informations QLDB dans CloudTrail

CloudTrail est activé sur votre compte Compte AWS lorsque vous créez le compte. Lorsqu'une activité prise en charge se produit dans QLDB, cette activité est enregistrée dans CloudTrail un événement avec d' Service AWS autres événements dans l'historique des événements. Vous pouvez consulter, rechercher et télécharger les événements récents dans votre Compte AWS. Pour plus d'informations, consultez la section Affichage des événements avec l'historique des CloudTrail événements.

Pour un enregistrement continu des événements de votre compte Compte AWS, y compris des événements pour QLDB, créez un parcours. Un suivi permet CloudTrail de fournir des fichiers journaux à un compartiment Amazon S3. Par défaut, lorsque vous créez un journal d'activité dans la console, il s'applique à toutes les régions Régions AWS. Le journal enregistre les événements de toutes les régions de la AWS partition et transmet les fichiers journaux au compartiment Amazon S3 que vous spécifiez. En outre, vous pouvez en configurer d'autres Services AWS pour analyser plus en détail les données d'événements collectées dans les CloudTrail journaux et agir en conséquence.

Pour plus d'informations, consultez les rubriques suivantes dans le AWS CloudTrail Guide de l'utilisateur :

- Présentation de la création d'un journal d'activité
- CloudTrail services et intégrations pris en charge

- Configuration des notifications Amazon SNS pour CloudTrail
- Réception de fichiers CloudTrail journaux provenant de plusieurs régions
- Réception de fichiers CloudTrail journaux provenant de plusieurs comptes

Toutes les actions de gestion des ressources QLDB et d'API de données non transactionnelles sont enregistrées et CloudTrail documentées dans la référence des API Amazon QLDB. Par exemple, les appels aux `CreateLedger``DescribeLedger`, et `DeleteLedger` les actions génèrent des entrées dans les fichiers CloudTrail journaux.

Chaque événement ou entrée de journal contient des informations sur la personne ayant initié la demande. Les informations relatives à l'identité permettent de déterminer les éléments suivants :

- Si la demande a été effectuée avec les informations d'identification utilisateur racine ou
- Si la demande a été effectuée avec des informations d'identification de sécurité temporaires pour un rôle ou un utilisateur fédéré
- Si la demande a été faite par un autre Service AWS

Pour plus d'informations, consultez l'élément CloudTrail UserIdentity.

Comprendre les entrées du fichier journal QLDB

Un suivi est une configuration qui permet de transmettre des événements sous forme de fichiers journaux à un compartiment Amazon S3 que vous spécifiez. CloudTrail les fichiers journaux contiennent une ou plusieurs entrées de journal. Un événement représente une demande unique provenant de n'importe quelle source et inclut des informations sur l'action demandée, la date et l'heure de l'action, les paramètres de la demande, etc. CloudTrail les fichiers journaux ne constituent pas une trace ordonnée des appels d'API publics, ils n'apparaissent donc pas dans un ordre spécifique.

L'exemple suivant montre une entrée de CloudTrail journal qui illustre ces actions :

- `CreateLedger`
- `DescribeLedger`
- `ListTagsForResource`
- `TagResource`
- `UntagResource`

- ListLedgers
- GetDigest
- GetBlock
- GetRevision
- ExportJournalToS3
- DescribeJournalS3Export
- ListJournalS3ExportsForLedger
- ListJournalS3Exports
- DeleteLedger


```
{
  "endTime": 1561497717208,
  "startTime": 1561497687254,
  "calls": [
    {
      "cloudtrailEvent": {
        "userIdentity": {
          "arn": "arn:aws:sts::123456789012:assumed-role/Admin/test-user"
        },
        "eventTime": "2019-06-25T21:21:27Z",
        "eventSource": "qldb.amazonaws.com",
        "eventName": "CreateLedger",
        "awsRegion": "us-east-2",
        "errorCode": null,
        "requestParameters": {
          "Name": "CloudtrailTest",
          "PermissionsMode": "ALLOW_ALL"
        },
        "responseElements": {
          "CreationDateTime": 1.561497687403E9,
          "Arn": "arn:aws:qldb:us-east-2:123456789012:ledger/CloudtrailTest",
          "State": "CREATING",
          "Name": "CloudtrailTest"
        },
        "requestID": "3135aec7-978f-11e9-b313-1dd92a14919e",
        "eventID": "bf703ff9-676f-41dd-be6f-5f666c9f7852",
        "readOnly": false,
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
```

```
      },
      "rawCloudtrailEvent": "{\"eventVersion\":\"1.05\",\"userIdentity\":{\"type
\":\"AssumedRole\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE:test-user\",\"arn
\":\"arn:aws:sts::123456789012:assumed-role/Admin/test-user\",\"accountId\":
\"123456789012\",\"accessKeyId\":\"AKIAI44QH8DHBEXAMPLE\",\"sessionContext\":
{\"attributes\":{\"mfaAuthenticated\":\"false\",\"creationDate\":\"2019-06-25T21:21:25Z
\"},\"sessionIssuer\":{\"type\":\"Role\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE\",
\"arn\":\"arn:aws:iam::123456789012:role/Admin\",\"accountId\":\"123456789012\",
\"userName\":\"Admin\"}}},\"eventTime\":\"2019-06-25T21:21:27Z\",\"eventSource
\":\"qldb.amazonaws.com\",\"eventName\":\"CreateLedger\",\"awsRegion\":\"us-
east-2\",\"sourceIPAddress\":\"192.0.2.01\",\"userAgent\":\"aws-internal/3 aws-
sdk-java/1.11.575 Mac_OS_X/10.13.6 Java_HotSpot(TM)_64-Bit_Server_VM/25.202-b08
 java/1.8.0_202 kotlin/1.3.21 vendor/Oracle_Corporation\",\"requestParameters\":
{\"Name\":\"CloudtrailTest\",\"PermissionsMode\":\"ALLOW_ALL\"},\"responseElements
\":{\"CreationDateTime\":1.561497687403E9,\"Arn\":\"arn:aws:qldb:us-
east-2:123456789012:ledger/CloudtrailTest\",\"State\":\"CREATING\",\"Name\":
\"CloudtrailTest\"},\"requestID\":\"3135aec7-978f-11e9-b313-1dd92a14919e\",\"eventID\":
\"bf703ff9-676f-41dd-be6f-5f666c9f7852\",\"readOnly\":false,\"eventType\":\"AwsApiCall
\",\"recipientAccountId\":\"123456789012\"}",
      "name": "CreateLedger",
      "request": [
        "com.amazonaws.services.qldb.model.CreateLedgerRequest",
        {
          "customRequestHeaders": null,
          "customQueryParameters": null,
          "name": "CloudtrailTest",
          "tags": null,
          "permissionsMode": "ALLOW_ALL"
        }
      ],
      "requestId": "3135aec7-978f-11e9-b313-1dd92a14919e"
    },
    {
      "cloudtrailEvent": {
        "userIdentity": {
          "arn": "arn:aws:sts::123456789012:assumed-role/Admin/test-user"
        },
        "eventTime": "2019-06-25T21:21:43Z",
        "eventSource": "qldb.amazonaws.com",
        "eventName": "DescribeLedger",
        "awsRegion": "us-east-2",
        "errorCode": null,
        "requestParameters": {
          "name": "CloudtrailTest"
```

```
        },
        "responseElements": null,
        "requestID": "3af51ba0-978f-11e9-8ae6-837dd17a19f8",
        "eventID": "be128e61-3e38-4503-83de-49fdc7fc0afb",
        "readOnly": true,
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
      },
      "rawCloudtrailEvent": "{\"eventVersion\":\"1.05\",\"userIdentity
\":{\"type\":\"AssumedRole\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE:test-
user\",\"arn\":\"arn:aws:sts::123456789012:assumed-role/Admin/test-user\",
\"accountId\":\"123456789012\",\"accessKeyId\":\"AKIAI44QH8DHBEXAMPLE\",
\"sessionContext\":{\"attributes\":{\"mfaAuthenticated\":\"false\",\"creationDate
\":\"2019-06-25T21:21:25Z\"},\"sessionIssuer\":{\"type\":\"Role\",\"principalId\":
\"AKIAIOSFODNN7EXAMPLE\",\"arn\":\"arn:aws:iam::123456789012:role/Admin\",\"accountId
\":\"123456789012\",\"userName\":\"Admin\"}}},\"eventTime\":\"2019-06-25T21:21:43Z
\",\"eventSource\":\"qldb.amazonaws.com\",\"eventName\":\"DescribeLedger\",
\"awsRegion\":\"us-east-2\",\"sourceIPAddress\":\"192.0.2.01\",\"userAgent\":
\"aws-internal/3 aws-sdk-java/1.11.575 Mac_OS_X/10.13.6 Java_HotSpot(TM)_64-
Bit_Server_VM/25.202-b08 java/1.8.0_202 kotlin/1.3.21 vendor/Oracle_Corporation
\",\"requestParameters\":{\"name\":\"CloudtrailTest\"},\"responseElements
\":null,\"requestID\":\"3af51ba0-978f-11e9-8ae6-837dd17a19f8\",\"eventID\":
\"be128e61-3e38-4503-83de-49fdc7fc0afb\",\"readOnly\":true,\"eventType\":\"AwsApiCall
\",\"recipientAccountId\":\"123456789012\"}",
      "name": "DescribeLedger",
      "request": [
        "com.amazonaws.services.qldb.model.DescribeLedgerRequest",
        {
          "customRequestHeaders": null,
          "customQueryParameters": null,
          "name": "CloudtrailTest"
        }
      ],
      "requestId": "3af51ba0-978f-11e9-8ae6-837dd17a19f8"
    },
    {
      "cloudtrailEvent": {
        "userIdentity": {
          "arn": "arn:aws:sts::123456789012:assumed-role/Admin/test-user"
        },
        "eventTime": "2019-06-25T21:21:44Z",
        "eventSource": "qldb.amazonaws.com",
        "eventName": "TagResource",
        "awsRegion": "us-east-2",
```

```
        "errorCode": null,
        "requestParameters": {
          "resourceArn": "arn%3Aaws%3Aqldb%3Aus-east-2%3A123456789012%3Aledger
%2FCloudtrailTest",
          "Tags": {
            "TagKey": "TagValue"
          }
        },
        "responseElements": null,
        "requestID": "3b1d6371-978f-11e9-916c-b7d64ec76521",
        "eventID": "6101c94a-7683-4431-812b-9a91afb8c849",
        "readOnly": false,
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
      },
      "rawCloudtrailEvent": "{\"eventVersion\":\"1.05\",\"userIdentity\":{\"type
\":\"AssumedRole\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE:test-user\",\"arn
\":\"arn:aws:sts::123456789012:assumed-role/Admin/test-user\",\"accountId\":
\"123456789012\",\"accessKeyId\":\"AKIAI44QH8DHBEXAMPLE\",\"sessionContext\":
{\"attributes\":{\"mfaAuthenticated\":\"false\",\"creationDate\":\"2019-06-25T21:21:25Z
\"},\"sessionIssuer\":{\"type\":\"Role\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE\",
\"arn\":\"arn:aws:iam::123456789012:role/Admin\",\"accountId\":\"123456789012\",
\"userName\":\"Admin\"}}},\"eventTime\":\"2019-06-25T21:21:44Z\",\"eventSource\":
\"qldb.amazonaws.com\",\"eventName\":\"TagResource\",\"awsRegion\":\"us-east-2\",
\"sourceIPAddress\":\"192.0.2.01\",\"userAgent\":\"aws-internal/3 aws-sdk-java/1.11.575
 Mac_OS_X/10.13.6 Java_HotSpot(TM)_64-Bit_Server_VM/25.202-b08 java/1.8.0_202
 kotlin/1.3.21 vendor/Oracle_Corporation\",\"requestParameters\":{\"resourceArn\":\"arn
%3Aaws%3Aqldb%3Aus-east-2%3A123456789012%3Aledger%2FCloudtrailTest\",\"Tags\":{\"TagKey
\":\"TagValue\"}},\"responseElements\":null,\"requestID\":\"3b1d6371-978f-11e9-916c-
b7d64ec76521\",\"eventID\":\"6101c94a-7683-4431-812b-9a91afb8c849\",\"readOnly\":false,
\"eventType\":\"AwsApiCall\",\"recipientAccountId\":\"123456789012\"}",
      "name": "TagResource",
      "request": [
        "com.amazonaws.services.qldb.model.TagResourceRequest",
        {
          "customRequestHeaders": null,
          "customQueryParameters": null,
          "resourceArn": "arn:aws:qldb:us-east-2:123456789012:ledger/CloudtrailTest",
          "tags": {
            "TagKey": "TagValue"
          }
        }
      ],
      "requestId": "3b1d6371-978f-11e9-916c-b7d64ec76521"
```

```
    },
    {
      "cloudtrailEvent": {
        "userIdentity": {
          "arn": "arn:aws:sts::123456789012:assumed-role/Admin/test-user"
        },
        "eventTime": "2019-06-25T21:21:44Z",
        "eventSource": "qldb.amazonaws.com",
        "eventName": "ListTagsForResource",
        "awsRegion": "us-east-2",
        "errorCode": null,
        "requestParameters": {
          "resourceArn": "arn%3Aaws%3Aqldb%3Aus-east-2%3A123456789012%3Aledger
%2FCloudtrailTest"
        },
        "responseElements": null,
        "requestID": "3b56c321-978f-11e9-8527-2517d5bfa8fd",
        "eventID": "375e57d7-cf94-495a-9a48-ac2192181c02",
        "readOnly": true,
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
      },
      "rawCloudtrailEvent": "{\"eventVersion\":\"1.05\",\"userIdentity
\":{\"type\":\"AssumedRole\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE:test-
user\",\"arn\":\"arn:aws:sts::123456789012:assumed-role/Admin/test-user\",
\"accountId\":\"123456789012\",\"accessKeyId\":\"AKIAI44QH8DHBEXAMPLE\",
\"sessionContext\":{\"attributes\":{\"mfaAuthenticated\":\"false\",\"creationDate
\":\"2019-06-25T21:21:25Z\"},\"sessionIssuer\":{\"type\":\"Role\",\"principalId
\":\"AKIAIOSFODNN7EXAMPLE\",\"arn\":\"arn:aws:iam::123456789012:role/Admin
\",\"accountId\":\"123456789012\",\"userName\":\"Admin\"}}},\"eventTime\":
\"2019-06-25T21:21:44Z\",\"eventSource\":\"qldb.amazonaws.com\",\"eventName
\":\"ListTagsForResource\",\"awsRegion\":\"us-east-2\",\"sourceIPAddress\":
\"192.0.2.01\",\"userAgent\":\"aws-internal/3 aws-sdk-java/1.11.575 Mac_OS_X/10.13.6
 Java_HotSpot(TM)_64-Bit_Server_VM/25.202-b08 java/1.8.0_202 kotlin/1.3.21 vendor/
Oracle_Corporation\",\"requestParameters\":{\"resourceArn\":\"arn%3Aaws%3Aqldb
%3Aus-east-2%3A123456789012%3Aledger%2FCloudtrailTest\"},\"responseElements\":null,
\"requestID\":\"3b56c321-978f-11e9-8527-2517d5bfa8fd\",\"eventID\":\"375e57d7-
cf94-495a-9a48-ac2192181c02\",\"readOnly\":true,\"eventType\":\"AwsApiCall\",
\"recipientAccountId\":\"123456789012\"}",
      "name": "ListTagsForResource",
      "request": [
        "com.amazonaws.services.qldb.model.ListTagsForResourceRequest",
        {
          "customRequestHeaders": null,
```

```
          "customQueryParameters": null,
          "resourceArn": "arn:aws:qldb:us-east-2:123456789012:ledger/CloudtrailTest"
        }
      ],
      "requestId": "3b56c321-978f-11e9-8527-2517d5bfa8fd"
    },
    {
      "cloudtrailEvent": {
        "userIdentity": {
          "arn": "arn:aws:sts::123456789012:assumed-role/Admin/test-user"
        },
        "eventTime": "2019-06-25T21:21:44Z",
        "eventSource": "qldb.amazonaws.com",
        "eventName": "UntagResource",
        "awsRegion": "us-east-2",
        "errorCode": null,
        "requestParameters": {
          "tagKeys": "TagKey",
          "resourceArn": "arn%3Aaws%3Aqldb%3Aus-east-2%3A123456789012%3Aledger
%2FCloudtrailTest"
        },
        "responseElements": null,
        "requestID": "3b87e59b-978f-11e9-8b9a-bb6dc3a800a9",
        "eventID": "bcdcdca3-699f-4363-b092-88242780406f",
        "readOnly": false,
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
      },
      "rawCloudtrailEvent": "{\"eventVersion\":\"1.05\",\"userIdentity\":{\"type
\":\"AssumedRole\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE:test-user\",\"arn
\":\"arn:aws:sts::123456789012:assumed-role/Admin/test-user\",\"accountId\":
\"123456789012\",\"accessKeyId\":\"AKIAI44QH8DHBEXAMPLE\",\"sessionContext\":
{\"attributes\":{\"mfaAuthenticated\":\"false\",\"creationDate\":\"2019-06-25T21:21:25Z
\"},\"sessionIssuer\":{\"type\":\"Role\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE\",
\"arn\":\"arn:aws:iam::123456789012:role/Admin\",\"accountId\":\"123456789012\",
\"userName\":\"Admin\"}}},\"eventTime\":\"2019-06-25T21:21:44Z\",\"eventSource\":
\"qldb.amazonaws.com\",\"eventName\":\"UntagResource\",\"awsRegion\":\"us-east-2\",
\"sourceIPAddress\":\"192.0.2.01\",\"userAgent\":\"aws-internal/3 aws-sdk-java/1.11.575
 Mac_OS_X/10.13.6 Java_HotSpot(TM)_64-Bit_Server_VM/25.202-b08 java/1.8.0_202
 kotlin/1.3.21 vendor/Oracle_Corporation\",\"requestParameters\":{\"tagKeys\":
\"TagKey\",\"resourceArn\":\"arn%3Aaws%3Aqldb%3Aus-east-2%3A123456789012%3Aledger
%2FCloudtrailTest\"},\"responseElements\":null,\"requestID\":\"3b87e59b-978f-11e9-8b9a-
bb6dc3a800a9\",\"eventID\":\"bcdcdca3-699f-4363-b092-88242780406f\",\"readOnly\":false,
\"eventType\":\"AwsApiCall\",\"recipientAccountId\":\"123456789012\"}",
```

```
      "name": "UntagResource",
      "request": [
        "com.amazonaws.services.qldb.model.UntagResourceRequest",
        {
          "customRequestHeaders": null,
          "customQueryParameters": null,
          "resourceArn": "arn:aws:qldb:us-east-2:123456789012:ledger/CloudtrailTest",
          "tagKeys": [
            "TagKey"
          ]
        }
      ],
      "requestId": "3b87e59b-978f-11e9-8b9a-bb6dc3a800a9"
    },
    {
      "cloudtrailEvent": {
        "userIdentity": {
          "arn": "arn:aws:sts::123456789012:assumed-role/Admin/test-user"
        },
        "eventTime": "2019-06-25T21:21:44Z",
        "eventSource": "qldb.amazonaws.com",
        "eventName": "ListLedgers",
        "awsRegion": "us-east-2",
        "errorCode": null,
        "requestParameters": null,
        "responseElements": null,
        "requestID": "3bafb877-978f-11e9-a6de-dbe6464b9dec",
        "eventID": "6ebe7d49-af59-4f29-aaa2-beffe536e20c",
        "readOnly": true,
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
      },
      "rawCloudtrailEvent": "{\"eventVersion\":\"1.05\",\"userIdentity\":{\"type
\":\"AssumedRole\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE:test-user\",\"arn
\":\"arn:aws:sts::123456789012:assumed-role/Admin/test-user\",\"accountId\":
\"123456789012\",\"accessKeyId\":\"AKIAI44QH8DHBEXAMPLE\",\"sessionContext\":
{\"attributes\":{\"mfaAuthenticated\":\"false\",\"creationDate\":\"2019-06-25T21:21:25Z
\"},\"sessionIssuer\":{\"type\":\"Role\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE\",
\"arn\":\"arn:aws:iam::123456789012:role/Admin\",\"accountId\":\"123456789012\",
\"userName\":\"Admin\"}}},\"eventTime\":\"2019-06-25T21:21:44Z\",\"eventSource
\":\"qldb.amazonaws.com\",\"eventName\":\"ListLedgers\",\"awsRegion\":\"us-
east-2\",\"sourceIPAddress\":\"192.0.2.01\",\"userAgent\":\"aws-internal/3 aws-
sdk-java/1.11.575 Mac_OS_X/10.13.6 Java_HotSpot(TM)_64-Bit_Server_VM/25.202-b08
 java/1.8.0_202 kotlin/1.3.21 vendor/Oracle_Corporation\",\"requestParameters\":null,
```

\"responseElements\":null,\"requestID\":\"3bafb877-978f-11e9-a6de-dbe6464b9dec\",
\"eventID\":\"6ebe7d49-af59-4f29-aaa2-beffe536e20c\",\"readOnly\":true,\"eventType\":
\"AwsApiCall\",\"recipientAccountId\":\"123456789012\"}",

```
      "name": "ListLedgers",
      "request": [
        "com.amazonaws.services.qldb.model.ListLedgersRequest",
        {
          "customRequestHeaders": null,
          "customQueryParameters": null,
          "maxResults": null,
          "nextToken": null
        }
      ],
      "requestId": "3bafb877-978f-11e9-a6de-dbe6464b9dec"
    },
    {
      "cloudtrailEvent": {
        "userIdentity": {
          "arn": "arn:aws:sts::123456789012:assumed-role/Admin/test-user"
        },
        "eventTime": "2019-06-25T21:21:49Z",
        "eventSource": "qldb.amazonaws.com",
        "eventName": "GetDigest",
        "awsRegion": "us-east-2",
        "errorCode": null,
        "requestParameters": {
          "name": "CloudtrailTest"
        },
        "responseElements": null,
        "requestID": "3cddd8a1-978f-11e9-a6de-dbe6464b9dec",
        "eventID": "a5cb60db-e6c5-4f5e-a5fc-0712249622b3",
        "readOnly": true,
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
      },
      "rawCloudtrailEvent": "{\"eventVersion\":\"1.05\",\"userIdentity\":{\"type
\":\"AssumedRole\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE:test-user\",\"arn
\":\"arn:aws:sts::123456789012:assumed-role/Admin/test-user\",\"accountId\":
\"123456789012\",\"accessKeyId\":\"AKIAI44QH8DHBEXAMPLE\",\"sessionContext\":
{\"attributes\":{\"mfaAuthenticated\":\"false\",\"creationDate\":\"2019-06-25T21:21:25Z
\"},\"sessionIssuer\":{\"type\":\"Role\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE\",
\"arn\":\"arn:aws:iam::123456789012:role/Admin\",\"accountId\":\"123456789012\",
\"userName\":\"Admin\"}}},\"eventTime\":\"2019-06-25T21:21:49Z\",\"eventSource\":
\"qldb.amazonaws.com\",\"eventName\":\"GetDigest\",\"awsRegion\":\"us-east-2\",
```

```
\"sourceIPAddress\":\"192.0.2.01\",\"userAgent\":\"aws-internal/3 aws-sdk-java/1.11.575
 Mac_OS_X/10.13.6 Java_HotSpot(TM)_64-Bit_Server_VM/25.202-b08 java/1.8.0_202
 kotlin/1.3.21 vendor/Oracle_Corporation\",\"requestParameters\":{\"name\":
\"CloudtrailTest\"},\"responseElements\":null,\"requestID\":\"3cddd8a1-978f-11e9-a6de-
dbe6464b9dec\",\"eventID\":\"a5cb60db-e6c5-4f5e-a5fc-0712249622b3\",\"readOnly\":true,
\"eventType\":\"AwsApiCall\",\"recipientAccountId\":\"123456789012\"}",
      "name": "GetDigest",
      "request": [
        "com.amazonaws.services.qldb.model.GetDigestRequest",
        {
          "customRequestHeaders": null,
          "customQueryParameters": null,
          "name": "CloudtrailTest",
          "digestTipAddress": null
        }
      ],
      "requestId": "3cddd8a1-978f-11e9-a6de-dbe6464b9dec"
    },
    {
      "cloudtrailEvent": {
        "userIdentity": {
          "arn": "arn:aws:sts::123456789012:assumed-role/Admin/test-user"
        },
        "eventTime": "2019-06-25T21:21:50Z",
        "eventSource": "qldb.amazonaws.com",
        "eventName": "GetBlock",
        "awsRegion": "us-east-2",
        "errorCode": null,
        "requestParameters": {
          "BlockAddress": {
            "IonText": "{strandId:\"2P2nsG3K2RwHQccUbnAMAj\",sequenceNo:0}"
          },
          "name": "CloudtrailTest",
          "DigestTipAddress": {
            "IonText": "{strandId:\"2P2nsG3K2RwHQccUbnAMAj\",sequenceNo:0}"
          }
        },
        "responseElements": null,
        "requestID": "3eaea09f-978f-11e9-bdc2-c1e55368155e",
        "eventID": "1f7da83f-d829-4e35-953d-30b925ceee66",
        "readOnly": true,
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
      },
```

```
      "rawCloudtrailEvent": "{\"eventVersion\":\"1.05\",\"userIdentity\":{\"type
\":\"AssumedRole\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE:test-user\",\"arn
\":\"arn:aws:sts::123456789012:assumed-role/Admin/test-user\",\"accountId\":
\"123456789012\",\"accessKeyId\":\"AKIAI44QH8DHBEXAMPLE\",\"sessionContext\":
{\"attributes\":{\"mfaAuthenticated\":\"false\",\"creationDate\":\"2019-06-25T21:21:25Z
\"},\"sessionIssuer\":{\"type\":\"Role\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE\",
\"arn\":\"arn:aws:iam::123456789012:role/Admin\",\"accountId\":\"123456789012\",
\"userName\":\"Admin\"}}},\"eventTime\":\"2019-06-25T21:21:50Z\",\"eventSource
\":\"qldb.amazonaws.com\",\"eventName\":\"GetBlock\",\"awsRegion\":\"us-east-2\",
\"sourceIPAddress\":\"192.0.2.01\",\"userAgent\":\"aws-internal/3 aws-sdk-java/1.11.575
 Mac_OS_X/10.13.6 Java_HotSpot(TM)_64-Bit_Server_VM/25.202-b08 java/1.8.0_202
 kotlin/1.3.21 vendor/Oracle_Corporation\",\"requestParameters\":{\"BlockAddress
\":{\"IonText\":\"{strandId:\\\"2P2nsG3K2RwHQccUbnAMAj\\\",sequenceNo:0}\"},
\"name\":\"CloudtrailTest\",\"DigestTipAddress\":{\"IonText\":\"{strandId:
\\\"2P2nsG3K2RwHQccUbnAMAj\\\",sequenceNo:0}\"}},\"responseElements\":null,
\"requestID\":\"3eaea09f-978f-11e9-bdc2-c1e55368155e\",\"eventID\":\"1f7da83f-
d829-4e35-953d-30b925ceee66\",\"readOnly\":true,\"eventType\":\"AwsApiCall\",
\"recipientAccountId\":\"123456789012\"}",
      "name": "GetBlock",
      "request": [
        "com.amazonaws.services.qldb.model.GetBlockRequest",
        {
          "customRequestHeaders": null,
          "customQueryParameters": null,
          "name": "CloudtrailTest",
          "blockAddress": {
            "ionText": "{strandId:\"2P2nsG3K2RwHQccUbnAMAj\",sequenceNo:0}"
          },
          "digestTipAddress": {
            "ionText": "{strandId:\"2P2nsG3K2RwHQccUbnAMAj\",sequenceNo:0}"
          }
        }
      ],
      "requestId": "3eaea09f-978f-11e9-bdc2-c1e55368155e"
    },
    {
      "cloudtrailEvent": {
        "userIdentity": {
          "arn": "arn:aws:sts::123456789012:assumed-role/Admin/test-user"
        },
        "eventTime": "2019-06-25T21:21:55Z",
        "eventSource": "qldb.amazonaws.com",
        "eventName": "GetRevision",
        "awsRegion": "us-east-2",
```

```
        "errorCode": null,
        "requestParameters": {
          "BlockAddress": {
            "IonText": "{strandId:\"2P2nsG3K2RwHQccUbnAMAj\",sequenceNo:1}"
          },
          "name": "CloudtrailTest",
          "DocumentId": "8UyXvDw6ApoFfVOA2HPfUE",
          "DigestTipAddress": {
            "IonText": "{strandId:\"2P2nsG3K2RwHQccUbnAMAj\",sequenceNo:1}"
          }
        },
        "responseElements": null,
        "requestID": "41e19139-978f-11e9-aaed-dfe1dafe37ab",
        "eventID": "43bf2661-5046-41ec-a1d3-87706954aa10",
        "readOnly": true,
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
      },
      "rawCloudtrailEvent": "{\"eventVersion\":\"1.05\",\"userIdentity\":{\"type
\":\"AssumedRole\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE:test-user\",\"arn
\":\"arn:aws:sts::123456789012:assumed-role/Admin/test-user\",\"accountId\":
\"123456789012\",\"accessKeyId\":\"AKIAI44QH8DHBEXAMPLE\",\"sessionContext\":
{\"attributes\":{\"mfaAuthenticated\":\"false\",\"creationDate\":\"2019-06-25T21:21:25Z
\"},\"sessionIssuer\":{\"type\":\"Role\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE\",
\"arn\":\"arn:aws:iam::123456789012:role/Admin\",\"accountId\":\"123456789012\",
\"userName\":\"Admin\"}}},\"eventTime\":\"2019-06-25T21:21:55Z\",\"eventSource\":
\"qldb.amazonaws.com\",\"eventName\":\"GetRevision\",\"awsRegion\":\"us-east-2\",
\"sourceIPAddress\":\"192.0.2.01\",\"userAgent\":\"aws-internal/3 aws-sdk-java/1.11.575
 Mac_OS_X/10.13.6 Java_HotSpot(TM)_64-Bit_Server_VM/25.202-b08 java/1.8.0_202
 kotlin/1.3.21 vendor/Oracle_Corporation\",\"requestParameters\":{\"BlockAddress
\":{\"IonText\":\"{strandId:\\\"2P2nsG3K2RwHQccUbnAMAj\\\",sequenceNo:1}\"},\"name
\":\"CloudtrailTest\",\"DocumentId\":\"8UyXvDw6ApoFfVOA2HPfUE\",\"DigestTipAddress
\":{\"IonText\":\"{strandId:\\\"2P2nsG3K2RwHQccUbnAMAj\\\",sequenceNo:1}\"}},
\"responseElements\":null,\"requestID\":\"41e19139-978f-11e9-aaed-dfe1dafe37ab\",
\"eventID\":\"43bf2661-5046-41ec-a1d3-87706954aa10\",\"readOnly\":true,\"eventType\":
\"AwsApiCall\",\"recipientAccountId\":\"123456789012\"}",
      "name": "GetRevision",
      "request": [
        "com.amazonaws.services.qldb.model.GetRevisionRequest",
        {
          "customRequestHeaders": null,
          "customQueryParameters": null,
          "name": "CloudtrailTest",
          "blockAddress": {
```

```
            "ionText": "{strandId:\"2P2nsG3K2RwHQccUbnAMAj\",sequenceNo:1}"
          },
          "documentId": "8UyXvDw6ApoFfVOA2HPfUE",
          "digestTipAddress": {
            "ionText": "{strandId:\"2P2nsG3K2RwHQccUbnAMAj\",sequenceNo:1}"
          }
        }
      ],
      "requestId": "41e19139-978f-11e9-aaed-dfe1dafe37ab"
    },
    {
      "cloudtrailEvent": {
        "userIdentity": {
          "arn": "arn:aws:sts::123456789012:assumed-role/Admin/test-user"
        },
        "eventTime": "2019-06-25T21:21:56Z",
        "eventSource": "qldb.amazonaws.com",
        "eventName": "ExportJournalToS3",
        "awsRegion": "us-east-2",
        "errorCode": null,
        "requestParameters": {
          "InclusiveStartTime": 1.561497687254E9,
          "name": "CloudtrailTest",
          "S3ExportConfiguration": {
            "Bucket": "cloudtrailtests-123456789012-us-east-2",
            "Prefix": "CloudtrailTestsJournalExport",
            "EncryptionConfiguration": {
              "ObjectEncryptionType": "SSE_S3"
            }
          },
          "ExclusiveEndTime": 1.561497715795E9
        },
        "responseElements": {
          "ExportId": "BabQhsmJRYDCGMnA2xYBDG"
        },
        "requestID": "423815f8-978f-11e9-afcf-55f7d0f3583d",
        "eventID": "1b5abdc4-52fa-435f-857e-8995ef7a19b7",
        "readOnly": false,
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
      },
      "rawCloudtrailEvent": "{\"eventVersion\":\"1.05\",\"userIdentity\":{\"type
\":\"AssumedRole\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE:test-user\",\"arn
\":\"arn:aws:sts::123456789012:assumed-role/Admin/test-user\",\"accountId\":
```

\"123456789012\",\"accessKeyId\":\"AKIAI44QH8DHBEXAMPLE\",\"sessionContext\":
{\"attributes\":{\"mfaAuthenticated\":\"false\",\"creationDate\":\"2019-06-25T21:21:25Z
\"},\"sessionIssuer\":{\"type\":\"Role\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE\",
\"arn\":\"arn:aws:iam::123456789012:role/Admin\",\"accountId\":\"123456789012\",
\"userName\":\"Admin\"}}},\"eventTime\":\"2019-06-25T21:21:56Z\",\"eventSource\":
\"qldb.amazonaws.com\",\"eventName\":\"ExportJournalToS3\",\"awsRegion\":\"us-east-2\",
\"sourceIPAddress\":\"192.0.2.01\",\"userAgent\":\"aws-internal/3 aws-sdk-java/1.11.575
 Mac_OS_X/10.13.6 Java_HotSpot(TM)_64-Bit_Server_VM/25.202-b08 java/1.8.0_202
 kotlin/1.3.21 vendor/Oracle_Corporation\",\"requestParameters\":{\"InclusiveStartTime
\":1.561497687254E9,\"name\":\"CloudtrailTest\",\"S3ExportConfiguration\":{\"Bucket\":
\"cloudtrailtests-123456789012-us-east-2\",\"Prefix\":\"CloudtrailTestsJournalExport\",
\"EncryptionConfiguration\":{\"ObjectEncryptionType\":\"SSE_S3\"}},\"ExclusiveEndTime
\":1.561497715795E9},\"responseElements\":{\"ExportId\":\"BabQhsmJRYDCGMnA2xYBDG
\"},\"requestID\":\"423815f8-978f-11e9-afcf-55f7d0f3583d\",\"eventID\":
\"1b5abdc4-52fa-435f-857e-8995ef7a19b7\",\"readOnly\":false,\"eventType\":\"AwsApiCall
\",\"recipientAccountId\":\"123456789012\"}",
      "name": "ExportJournalToS3",
      "request": [
        "com.amazonaws.services.qldb.model.ExportJournalToS3Request",
        {
          "customRequestHeaders": null,
          "customQueryParameters": null,
          "name": "CloudtrailTest",
          "inclusiveStartTime": 1561497687254,
          "exclusiveEndTime": 1561497715795,
          "s3ExportConfiguration": {
            "bucket": "cloudtrailtests-123456789012-us-east-2",
            "prefix": "CloudtrailTestsJournalExport",
            "encryptionConfiguration": {
              "objectEncryptionType": "SSE_S3",
              "kmsKeyArn": null
            }
          }
        }
      ],
      "requestId": "423815f8-978f-11e9-afcf-55f7d0f3583d"
    },
    {
      "cloudtrailEvent": {
        "userIdentity": {
          "arn": "arn:aws:sts::123456789012:assumed-role/Admin/test-user"
        },
        "eventTime": "2019-06-25T21:21:56Z",
        "eventSource": "qldb.amazonaws.com",

```
        "eventName": "DescribeJournalS3Export",
        "awsRegion": "us-east-2",
        "errorCode": null,
        "requestParameters": {
          "name": "CloudtrailTest",
          "exportId": "BabQhsmJRYDCGMnA2xYBDG"
        },
        "responseElements": null,
        "requestID": "427ebbbc-978f-11e9-8888-e9894c9c4bb9",
        "eventID": "ca8ffc88-16ff-45f5-9042-d94fadb389c3",
        "readOnly": true,
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
      },
      "rawCloudtrailEvent": "{\"eventVersion\":\"1.05\",\"userIdentity\":{\"type
\":\"AssumedRole\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE:test-user\",\"arn
\":\"arn:aws:sts::123456789012:assumed-role/Admin/test-user\",\"accountId\":
\"123456789012\",\"accessKeyId\":\"AKIAI44QH8DHBEXAMPLE\",\"sessionContext\":
{\"attributes\":{\"mfaAuthenticated\":\"false\",\"creationDate\":\"2019-06-25T21:21:25Z
\"},\"sessionIssuer\":{\"type\":\"Role\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE\",
\"arn\":\"arn:aws:iam::123456789012:role/Admin\",\"accountId\":\"123456789012\",
\"userName\":\"Admin\"}}},\"eventTime\":\"2019-06-25T21:21:56Z\",\"eventSource\":
\"qldb.amazonaws.com\",\"eventName\":\"DescribeJournalS3Export\",\"awsRegion\":
\"us-east-2\",\"sourceIPAddress\":\"192.0.2.01\",\"userAgent\":\"aws-internal/3
 aws-sdk-java/1.11.575 Mac_OS_X/10.13.6 Java_HotSpot(TM)_64-Bit_Server_VM/25.202-b08
 java/1.8.0_202 kotlin/1.3.21 vendor/Oracle_Corporation\",\"requestParameters\":{\"name
\":\"CloudtrailTest\",\"exportId\":\"BabQhsmJRYDCGMnA2xYBDG\"},\"responseElements
\":null,\"requestID\":\"427ebbbc-978f-11e9-8888-e9894c9c4bb9\",\"eventID\":
\"ca8ffc88-16ff-45f5-9042-d94fadb389c3\",\"readOnly\":true,\"eventType\":\"AwsApiCall
\",\"recipientAccountId\":\"123456789012\"}",
      "name": "DescribeJournalS3Export",
      "request": [
        "com.amazonaws.services.qldb.model.DescribeJournalS3ExportRequest",
        {
          "customRequestHeaders": null,
          "customQueryParameters": null,
          "name": "CloudtrailTest",
          "exportId": "BabQhsmJRYDCGMnA2xYBDG"
        }
      ],
      "requestId": "427ebbbc-978f-11e9-8888-e9894c9c4bb9"
    },
    {
      "cloudtrailEvent": {
```

```
        "userIdentity": {
          "arn": "arn:aws:sts::123456789012:assumed-role/Admin/test-user"
        },
        "eventTime": "2019-06-25T21:21:56Z",
        "eventSource": "qldb.amazonaws.com",
        "eventName": "ListJournalS3ExportsForLedger",
        "awsRegion": "us-east-2",
        "errorCode": null,
        "requestParameters": {
          "name": "CloudtrailTest"
        },
        "responseElements": null,
        "requestID": "429ca40c-978f-11e9-8c4b-d13a8018a286",
        "eventID": "34f0e76b-58a5-45be-881c-786d22e34e96",
        "readOnly": true,
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
      },
      "rawCloudtrailEvent": "{\"eventVersion\":\"1.05\",\"userIdentity\":{\"type
\":\"AssumedRole\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE:test-user\",\"arn
\":\"arn:aws:sts::123456789012:assumed-role/Admin/test-user\",\"accountId\":
\"123456789012\",\"accessKeyId\":\"AKIAI44QH8DHBEXAMPLE\",\"sessionContext\":
{\"attributes\":{\"mfaAuthenticated\":\"false\",\"creationDate\":\"2019-06-25T21:21:25Z
\"},\"sessionIssuer\":{\"type\":\"Role\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE\",
\"arn\":\"arn:aws:iam::123456789012:role/Admin\",\"accountId\":\"123456789012\",
\"userName\":\"Admin\"}}},\"eventTime\":\"2019-06-25T21:21:56Z\",\"eventSource
\":\"qldb.amazonaws.com\",\"eventName\":\"ListJournalS3ExportsForLedger\",
\"awsRegion\":\"us-east-2\",\"sourceIPAddress\":\"192.0.2.01\",\"userAgent\":
\"aws-internal/3 aws-sdk-java/1.11.575 Mac_OS_X/10.13.6 Java_HotSpot(TM)_64-
Bit_Server_VM/25.202-b08 java/1.8.0_202 kotlin/1.3.21 vendor/Oracle_Corporation
\",\"requestParameters\":{\"name\":\"CloudtrailTest\"},\"responseElements
\":null,\"requestID\":\"429ca40c-978f-11e9-8c4b-d13a8018a286\",\"eventID\":
\"34f0e76b-58a5-45be-881c-786d22e34e96\",\"readOnly\":true,\"eventType\":\"AwsApiCall
\",\"recipientAccountId\":\"123456789012\"}",
      "name": "ListJournalS3ExportsForLedger",
      "request": [
        "com.amazonaws.services.qldb.model.ListJournalS3ExportsForLedgerRequest",
        {
          "customRequestHeaders": null,
          "customQueryParameters": null,
          "name": "CloudtrailTest",
          "maxResults": null,
          "nextToken": null
        }
```

```
      ],
      "requestId": "429ca40c-978f-11e9-8c4b-d13a8018a286"
    },
    {
      "cloudtrailEvent": {
        "userIdentity": {
          "arn": "arn:aws:sts::123456789012:assumed-role/Admin/test-user"
        },
        "eventTime": "2019-06-25T21:21:56Z",
        "eventSource": "qldb.amazonaws.com",
        "eventName": "ListJournalS3Exports",
        "awsRegion": "us-east-2",
        "errorCode": null,
        "requestParameters": null,
        "responseElements": null,
        "requestID": "42cc1814-978f-11e9-befb-f5dbaa142118",
        "eventID": "4c24d7d6-810c-4cf4-884e-00482278b6ce",
        "readOnly": true,
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
      },
      "rawCloudtrailEvent": "{\"eventVersion\":\"1.05\",\"userIdentity\":{\"type
\":\"AssumedRole\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE:test-user\",\"arn
\":\"arn:aws:sts::123456789012:assumed-role/Admin/test-user\",\"accountId\":
\"123456789012\",\"accessKeyId\":\"AKIAI44QH8DHBEXAMPLE\",\"sessionContext\":
{\"attributes\":{\"mfaAuthenticated\":\"false\",\"creationDate\":\"2019-06-25T21:21:25Z
\"},\"sessionIssuer\":{\"type\":\"Role\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE\",
\"arn\":\"arn:aws:iam::123456789012:role/Admin\",\"accountId\":\"123456789012\",
\"userName\":\"Admin\"}}},\"eventTime\":\"2019-06-25T21:21:56Z\",\"eventSource\":
\"qldb.amazonaws.com\",\"eventName\":\"ListJournalS3Exports\",\"awsRegion\":\"us-
east-2\",\"sourceIPAddress\":\"192.0.2.01\",\"userAgent\":\"aws-internal/3 aws-
sdk-java/1.11.575 Mac_OS_X/10.13.6 Java_HotSpot(TM)_64-Bit_Server_VM/25.202-b08
 java/1.8.0_202 kotlin/1.3.21 vendor/Oracle_Corporation\",\"requestParameters\":null,
\"responseElements\":null,\"requestID\":\"42cc1814-978f-11e9-befb-f5dbaa142118\",
\"eventID\":\"4c24d7d6-810c-4cf4-884e-00482278b6ce\",\"readOnly\":true,\"eventType\":
\"AwsApiCall\",\"recipientAccountId\":\"123456789012\"}",
      "name": "ListJournalS3Exports",
      "request": [
        "com.amazonaws.services.qldb.model.ListJournalS3ExportsRequest",
        {
          "customRequestHeaders": null,
          "customQueryParameters": null,
          "maxResults": null,
          "nextToken": null
```

```
        }
      ],
      "requestId": "42cc1814-978f-11e9-befb-f5dbaa142118"
    },
    {
      "cloudtrailEvent": {
        "userIdentity": {
          "arn": "arn:aws:sts::123456789012:assumed-role/Admin/test-user"
        },
        "eventTime": "2019-06-25T21:21:57Z",
        "eventSource": "qldb.amazonaws.com",
        "eventName": "DeleteLedger",
        "awsRegion": "us-east-2",
        "errorCode": null,
        "requestParameters": {
          "name": "CloudtrailTest"
        },
        "responseElements": null,
        "requestID": "42f439b9-978f-11e9-8b2c-69ef598d66e9",
        "eventID": "429f5163-cba5-4d86-bd7e-f606e057c6cf",
        "readOnly": false,
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
      },
      "rawCloudtrailEvent": "{\"eventVersion\":\"1.05\",\"userIdentity\":{\"type
\":\"AssumedRole\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE:test-user\",\"arn
\":\"arn:aws:sts::123456789012:assumed-role/Admin/test-user\",\"accountId\":
\"123456789012\",\"accessKeyId\":\"AKIAI44QH8DHBEXAMPLE\",\"sessionContext\":
{\"attributes\":{\"mfaAuthenticated\":\"false\",\"creationDate\":\"2019-06-25T21:21:25Z
\"},\"sessionIssuer\":{\"type\":\"Role\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE\",
\"arn\":\"arn:aws:iam::123456789012:role/Admin\",\"accountId\":\"123456789012\",
\"userName\":\"Admin\"}}},\"eventTime\":\"2019-06-25T21:21:57Z\",\"eventSource
\":\"qldb.amazonaws.com\",\"eventName\":\"DeleteLedger\",\"awsRegion\":\"us-
east-2\",\"sourceIPAddress\":\"192.0.2.01\",\"userAgent\":\"aws-internal/3 aws-
sdk-java/1.11.575 Mac_OS_X/10.13.6 Java_HotSpot(TM)_64-Bit_Server_VM/25.202-b08
 java/1.8.0_202 kotlin/1.3.21 vendor/Oracle_Corporation\",\"requestParameters
\":{\"name\":\"CloudtrailTest\"},\"responseElements\":null,\"requestID\":
\"42f439b9-978f-11e9-8b2c-69ef598d66e9\",\"eventID\":\"429f5163-cba5-4d86-bd7e-
f606e057c6cf\",\"readOnly\":false,\"eventType\":\"AwsApiCall\",\"recipientAccountId\":
\"123456789012\"}",
      "name": "DeleteLedger",
      "request": [
        "com.amazonaws.services.qldb.model.DeleteLedgerRequest",
        {
```

```
          "customRequestHeaders": null,
          "customQueryParameters": null,
          "name": "CloudtrailTest"
        }
      ],
      "requestId": "42f439b9-978f-11e9-8b2c-69ef598d66e9"
    }
  ]
}
```

Validation de conformité pour Amazon QLDB

Des auditeurs tiers évaluent la sécurité et la conformité d'Amazon QLDB dans le cadre de AWS plusieurs programmes de conformité, notamment les suivants :

- System and Organization Controls (SOC)
- Payment Card Industry (PCI)
- Organisation internationale de normalisation (ISO)
- Programme de gestion et d'évaluation de la sécurité des systèmes d'information (ISMAP)
- Health Insurance Portability and Accountability Act (HIPAA)

> **Note**
>
> Il ne s'agit pas d'une liste exhaustive des certifications Amazon QLDB.

Pour savoir si un programme Services AWS de conformité Service AWS s'inscrit dans le champ d'application de programmes de conformité spécifiques, consultez Services AWS la section de conformité et sélectionnez le programme de conformité qui vous intéresse. Pour des informations générales, voir Programmes de AWS conformité Programmes AWS de .

Vous pouvez télécharger des rapports d'audit tiers à l'aide de AWS Artifact. Pour plus d'informations, voir Téléchargement de rapports dans AWS Artifact .

Votre responsabilité en matière de conformité lors de l'utilisation Services AWS est déterminée par la sensibilité de vos données, les objectifs de conformité de votre entreprise et les lois et réglementations applicables. AWS fournit les ressources suivantes pour faciliter la mise en conformité :

- Guides de démarrage rapide sur la sécurité et la conformité : ces guides de déploiement abordent les considérations architecturales et indiquent les étapes à suivre pour déployer des environnements de base axés sur AWS la sécurité et la conformité.
- Architecture axée sur la sécurité et la conformité HIPAA sur Amazon Web Services : ce livre blanc décrit comment les entreprises peuvent créer des applications AWS conformes à la loi HIPAA.

> **Note**
>
> Tous ne Services AWS sont pas éligibles à la loi HIPAA. Pour plus d'informations, consultez le HIPAA Eligible Services Reference.

- AWS Ressources de https://aws.amazon.com/compliance/resources/ de conformité — Cette collection de classeurs et de guides peut s'appliquer à votre secteur d'activité et à votre région.
- AWS Guides de conformité destinés aux clients — Comprenez le modèle de responsabilité partagée sous l'angle de la conformité. Les guides résument les meilleures pratiques en matière de sécurisation Services AWS et décrivent les directives relatives aux contrôles de sécurité dans de nombreux cadres (notamment le National Institute of Standards and Technology (NIST), le Payment Card Industry Security Standards Council (PCI) et l'Organisation internationale de normalisation (ISO)).
- Évaluation des ressources à l'aide des règles du guide du AWS Config développeur : le AWS Config service évalue dans quelle mesure les configurations de vos ressources sont conformes aux pratiques internes, aux directives du secteur et aux réglementations.
- AWS Security Hub— Cela Service AWS fournit une vue complète de votre état de sécurité interne AWS. Security Hub utilise des contrôles de sécurité pour évaluer vos ressources AWS et vérifier votre conformité par rapport aux normes et aux bonnes pratiques du secteur de la sécurité. Pour obtenir la liste des services et des contrôles pris en charge, consultez Référence des contrôles Security Hub.
- Amazon GuardDuty — Cela Service AWS détecte les menaces potentielles qui pèsent sur vos charges de travail Comptes AWS, vos conteneurs et vos données en surveillant votre environnement pour détecter toute activité suspecte et malveillante. GuardDuty peut vous aider à répondre à diverses exigences de conformité, telles que la norme PCI DSS, en répondant aux exigences de détection des intrusions imposées par certains cadres de conformité.
- AWS Audit Manager— Cela vous Service AWS permet d'auditer en permanence votre AWS utilisation afin de simplifier la gestion des risques et la conformité aux réglementations et aux normes du secteur.

Résilience dans Amazon QLDB

L'infrastructure AWS mondiale est construite autour Régions AWS de zones de disponibilité. Régions AWS fournissent plusieurs zones de disponibilité physiquement séparées et isolées, connectées par un réseau à faible latence, à haut débit et hautement redondant. Avec les zones de disponibilité, vous pouvez concevoir et exploiter des applications et des bases de données qui basculent automatiquement d'une zone de disponibilité à l'autre sans interruption. Les zones de disponibilité sont plus hautement disponibles, tolérantes aux pannes et évolutives que les infrastructures traditionnelles à un ou plusieurs centres de données.

Pour plus d'informations sur les zones de disponibilité Régions AWS et les zones de disponibilité, consultez la section Infrastructure AWS globale.

Durabilité du stockage

Le stockage des journaux QLDB permet une réplication synchrone vers plusieurs zones de disponibilité lors des validations de transactions. Cela garantit que même une défaillance complète de la zone de disponibilité du stockage des journaux ne compromettra pas l'intégrité des données ou la capacité de maintenir un service actif. En outre, le journal QLDB propose des archives asynchrones pour un stockage tolérant aux pannes. Cette fonctionnalité prend en charge la reprise après sinistre dans le cas très improbable d'une panne de stockage simultanée pour plusieurs zones de disponibilité.

Le stockage indexé QLDB est soutenu par une réplication vers plusieurs zones de disponibilité. Cela garantit que même une défaillance complète de la zone de disponibilité du stockage indexé ne compromet pas l'intégrité des données ou la capacité à maintenir un service actif.

Caractéristiques de durabilité des données

Outre l'infrastructure AWS mondiale, QLDB propose les fonctionnalités suivantes pour répondre à vos besoins en matière de résilience et de sauvegarde des données.

Fonctionnalités du service QLDB

Exportation de journaux à la demande

QLDB fournit une fonctionnalité d'exportation de journaux à la demande. Accédez au contenu de votre journal en exportant des blocs de journal de votre registre vers un compartiment Amazon S3. Vous pouvez utiliser ces données à diverses fins, telles que la conservation des données,

l'analyse et l'audit. Pour plus d'informations, consultez Exportation de données de journal depuis Amazon QLDB.

Sauvegarde et restauration

La restauration automatique pour les exportations n'est pas prise en charge pour le moment. L'exportation fournit une capacité de base pour créer une redondance de données supplémentaire à la fréquence que vous avez définie. Cependant, un scénario de restauration dépend de l'application, dans lequel les enregistrements exportés sont probablement réécrits dans un nouveau registre en utilisant la même méthode d'ingestion ou une méthode similaire.

QLDB ne fournit pas de fonction de sauvegarde dédiée ni de fonction de restauration associée pour le moment.

Streams de journaux

QLDB fournit également une fonctionnalité de flux de journal continu. Vous pouvez intégrer les flux de journaux QLDB à la plateforme de streaming Amazon Kinesis pour traiter les données des journaux en temps réel. Pour plus d'informations, consultez Diffusion en continu de données de journaux depuis Amazon QLDB.

Fonctionnalité de conception QLDB

QLDB est conçu pour résister à la corruption logique. Le journal QLDB est immuable, ce qui garantit que toutes les transactions validées sont conservées dans le journal. En outre, chaque modification de document validée est enregistrée, ce qui permet de point-in-timedétecter toute modification involontaire des données du registre.

QLDB ne fournit pas de fonctionnalité de restauration automatique pour les scénarios de corruption logique pour le moment.

Sécurité de l'infrastructure dans Amazon QLDB

En tant que service géré, Amazon QLDB est protégé par les procédures de sécurité AWS du réseau mondial décrites dans le livre blanc Amazon Web Services : présentation des processus de sécurité.

Vous utilisez des appels d'API AWS publiés pour accéder à QLDB via le réseau. Les clients doivent supporter le protocole TLS (Sécurité de la couche transport) 1.0 ou une version ultérieure. Nous recommandons TLS 1.2 ou version ultérieure. Les clients doivent aussi prendre en charge les suites de chiffrement PFS (Perfect Forward Secrecy) comme Ephemeral Diffie-Hellman (DHE) ou Elliptic

Curve Ephemeral Diffie-Hellman (ECDHE). La plupart des systèmes modernes tels que Java 7 et les versions ultérieures prennent en charge ces modes.

En outre, les demandes doivent être signées à l'aide d'informations d'identification programmatiques associées à un principal IAM. Vous pouvez également utiliser AWS Security Token Service (AWS STS) pour générer des informations d'identification de sécurité temporaires et signer les demandes.

Vous pouvez également utiliser un point de terminaison de cloud privé virtuel (VPC) pour QLDB. Les points de terminaison VPC d'interface permettent à vos ressources Amazon VPC d'utiliser leurs adresses IP privées pour accéder à QLDB sans accès à Internet public. Pour plus d'informations, consultez Accédez à Amazon QLDB à l'aide d'un point de terminaison d'interface ()AWS PrivateLink.

Accédez à Amazon QLDB à l'aide d'un point de terminaison d'interface ()AWS PrivateLink

Vous pouvez l'utiliser AWS PrivateLink pour créer une connexion privée entre votre VPC et Amazon QLDB. Vous pouvez accéder à QLDB comme s'il se trouvait dans votre VPC, sans utiliser de passerelle Internet, de périphérique NAT, de connexion VPN ou de connexion. AWS Direct Connect Les instances de votre VPC n'ont pas besoin d'adresses IP publiques pour accéder à QLDB.

Vous établissez cette connexion privée en créant un point de terminaison d'interface optimisé par AWS PrivateLink. Nous créons une interface réseau de point de terminaison dans chaque sous-réseau que vous activez pour le point de terminaison d'interface. Il s'agit d'interfaces réseau gérées par les demandeurs qui servent de point d'entrée pour le trafic destiné à QLDB.

Pour plus d'informations, consultez Accès aux Services AWS via AWS PrivateLink dans le Guide AWS PrivateLink .

Rubriques

- Considérations relatives à QLDB
- Création d'un point de terminaison d'interface pour QLDB
- Création d'une politique de point de terminaison pour votre point de terminaison d'interface
- Disponibilité des points de terminaison d'interface pour QLDB

Considérations relatives à QLDB

Avant de configurer un point de terminaison d'interface pour QLDB, consultez les considérations du guide.AWS PrivateLink

> **Note**
>
> QLDB prend uniquement en charge les appels vers l'API de données transactionnelles de la session QLDB via le point de terminaison de l'interface. Cette API inclut uniquement l'SendCommandopération. Dans le mode `STANDARD` autorisations d'un registre, vous pouvez contrôler les autorisations relatives à des actions partiQL spécifiques dans cette API.

Création d'un point de terminaison d'interface pour QLDB

Vous pouvez créer un point de terminaison d'interface pour QLDB à l'aide de la console Amazon VPC ou du (). AWS Command Line Interface AWS CLI Pour plus d'informations, consultez Création d'un point de terminaison d'interface dans le Guide AWS PrivateLink .

Créez un point de terminaison d'interface pour QLDB en utilisant le nom de service suivant :

```
com.amazonaws.region.qldb.session
```

Si vous activez le DNS privé pour le point de terminaison de l'interface, vous pouvez envoyer des demandes d'API à QLDB en utilisant son nom DNS régional par défaut. Par exemple, `session.qldb.us-east-1.amazonaws.com`.

Création d'une politique de point de terminaison pour votre point de terminaison d'interface

Une politique de point de terminaison est une ressource IAM que vous pouvez attacher à votre point de terminaison d'interface. La politique de point de terminaison par défaut autorise un accès complet à QLDB via le point de terminaison de l'interface. Pour contrôler l'accès autorisé à QLDB depuis votre VPC, associez une politique de point de terminaison personnalisée au point de terminaison de l'interface.

Une politique de point de terminaison spécifie les informations suivantes :

- Les principaux qui peuvent effectuer des actions (Comptes AWS utilisateurs et rôles).
- Les actions qui peuvent être effectuées.
- La ressource sur laquelle les actions peuvent être effectuées.

Pour plus d'informations, consultez Contrôle de l'accès aux services à l'aide de politiques de point de terminaison dans le Guide AWS PrivateLink .

Vous pouvez également utiliser le `Condition` champ dans une politique attachée à un utilisateur, un groupe ou un rôle pour autoriser l'accès uniquement à partir d'un point de terminaison d'interface spécifié. Lorsqu'elles sont utilisées conjointement, les politiques de point de terminaison et les politiques IAM peuvent restreindre l'accès à des actions QLDB spécifiques sur des registres spécifiques à un point de terminaison d'interface spécifié.

Exemple de politique de point de terminaison : restreindre l'accès à un registre QLDB spécifique

Voici un exemple de politique de point de terminaison personnalisée pour QLDB. Lorsque vous attachez cette politique au point de terminaison de votre interface, elle accorde l'accès à l'`SendCommand`action et aux actions partiQL en lecture seule à tous les principaux de la ressource de registre spécifiée. Dans cet exemple, le registre doit être en mode `STANDARD` autorisations.

Pour utiliser cette politique, remplacez *`us-east-1`*, 123456789012*`, et`* dans l'exemple par vos propres informations. *`myExampleLedger`*

```
{
   "Statement": [
      {
         "Sid": "QLDBSendCommandPermission",
         "Principal": "*",
         "Effect": "Allow",
         "Action": "qldb:SendCommand",
         "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
      },
      {
         "Sid": "QLDBPartiQLReadOnlyPermissions",
         "Principal": "*",
         "Effect": "Allow",
         "Action": [
            "qldb:PartiQLSelect",
            "qldb:PartiQLHistoryFunction"
         ],
         "Resource": [
            "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/*",
            "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/
information_schema/user_tables"
         ]
      }
```

```
   ]
}
```

Exemple de politique IAM : restreindre l'accès à un registre QLDB uniquement à partir d'un point de terminaison d'interface spécifique

Voici un exemple de stratégie basée sur l'identité IAM pour QLDB. Lorsque vous associez cette politique à un utilisateur, à un rôle ou à un groupe, elle autorise l'`SendCommand`accès à une ressource de registre uniquement à partir du point de terminaison d'interface spécifié.

Pour utiliser cette politique, remplacez us-east-1, 123456789012 et vpce-1a2b3c4d myExampleLedgerdans l'exemple par vos propres informations.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "AccessFromSpecificInterfaceEndpoint",
         "Effect": "Deny",
         "Action": "qldb:SendCommand",
         "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger",
         "Condition": {
            "StringNotEquals": {
               "aws:sourceVpce": "vpce-1a2b3c4d"
            }
         }
      }
   ]
}
```

Disponibilité des points de terminaison d'interface pour QLDB

Amazon QLDB prend en charge les points de terminaison d'interface avec des politiques partout Régions AWS où QLDB est disponible. Pour obtenir la liste complète des régions disponibles, consultez la section Points de terminaison et quotas Amazon QLDB dans le. Références générales AWS

Résolutions des problèmes liés à Amazon QLDB

Les sections suivantes fournissent une liste agrégée des erreurs courantes que vous pouvez rencontrer lors de l'utilisation d'Amazon QLDB, ainsi que des instructions pour les résoudre.

Pour obtenir des conseils de dépannage spécifiques à l'accès IAM, consultezRésolution des problèmes d'identité et d'accès à Amazon QLDB.

Pour connaître les meilleures pratiques relatives au réglage de vos instructions PartiQL, consultezOptimisation des performances des données.

Rubriques

Exécution de transactions à l'aide du pilote QLDB

Cette section répertorie les exceptions courantes que le pilote Amazon QLDB peut renvoyer lorsque vous l'utilisez pour exécuter des transactions partiQL sur un registre. Pour en savoir plus sur cette fonction, consultez Démarrage. Pour connaître les meilleures pratiques relatives à la configuration et à l'utilisation du pilote, reportez-vous à la sectionRecommandations du conducteur.

Chaque exception inclut le message d'erreur spécifique, suivi d'une brève description et de suggestions de solutions possibles.

CapacityExceededException

> Message : Capacité dépassée
>
> Amazon QLDB a rejeté la demande car elle dépassait la capacité de traitement du registre. QLDB applique une limite de mise à l'échelle interne par registre afin de préserver la santé et les performances du service. Cette limite varie en fonction de la taille de la charge de travail de chaque demande individuelle. Par exemple, une demande peut avoir une charge de travail accrue si elle effectue des transactions de données inefficaces, telles que des scans de tables résultant d'une requête qualifiée non indexée.

Nous vous recommandons d'attendre avant de réessayer la demande. Si votre application rencontre régulièrement cette exception, optimisez vos relevés et diminuez le taux et le volume des demandes que vous envoyez au grand livre. Parmi les exemples d'optimisation des relevés, citons l'exécution d'un moins grand nombre de relevés par transaction et le réglage des index de vos tables. Pour savoir comment optimiser les relevés et éviter les scans de tableaux, consultezOptimisation des performances des données.

Nous vous recommandons également d'utiliser la dernière version du pilote QLDB. Le pilote dispose d'une politique de nouvelle tentative par défaut qui utilise Exponential Backoff et Jitter pour réessayer automatiquement en cas d'exceptions telles que celle-ci. Le concept de recul exponentiel consiste à utiliser des temps d'attente progressivement plus longs entre les nouvelles tentatives pour des réponses d'erreur consécutives.

InvalidSessionException

Message : L'*`ID de transaction`* a expiré

Une transaction a dépassé sa durée de vie maximale. Une transaction peut être exécutée pendant 30 secondes au maximum avant d'être validée. Passé ce délai, tout travail effectué sur la transaction est rejeté et QLDB annule la session. Cette limite protège le client contre les fuites de sessions en commençant des transactions et en ne les validant ni en les annulant.

S'il s'agit d'une exception courante dans votre application, il est probable que les transactions prennent tout simplement trop de temps à s'exécuter. Si l'exécution des transactions prend plus de 30 secondes, optimisez vos relevés pour accélérer les transactions. Parmi les exemples d'optimisation des relevés, citons l'exécution d'un moins grand nombre de relevés par transaction et le réglage des index de vos tables. Pour plus d'informations, consultez Optimisation des performances des données.

InvalidSessionException

Message : L'*`sessionId`* a expiré

QLDB a annulé la session car elle dépassait sa durée de vie totale maximale. QLDB supprime les sessions après 13 à 17 minutes, quelle que soit une transaction active. Les sessions peuvent être perdues ou altérées pour diverses raisons, telles qu'une panne matérielle, une défaillance du réseau ou le redémarrage d'applications. Ainsi, QLDB impose une durée de vie maximale aux sessions afin de garantir la résilience du logiciel client en cas d'échec de session.

Si vous rencontrez cette exception, nous vous recommandons de créer une nouvelle session et de réessayer la transaction. Nous recommandons également d'utiliser la dernière version du pilote QLDB, qui gère le pool de sessions et son état au nom de l'application.

InvalidSessionException

Message : Aucune session de ce type

Le client a essayé d'effectuer une transaction avec QLDB en utilisant une session qui n'existe pas. En supposant que le client utilise une session qui existait auparavant, il se peut que la session n'existe plus pour l'une des raisons suivantes :

- Si une session est impliquée dans une défaillance interne du serveur (c'est-à-dire une erreur avec le code de réponse HTTP 500), QLDB peut choisir de supprimer complètement la session plutôt que d'autoriser le client à effectuer une transaction avec une session dont l'état est incertain. Ensuite, toute nouvelle tentative sur cette session échoue avec cette erreur.
- Les sessions expirées sont finalement oubliées par QLDB. Ensuite, toute tentative de continuer à utiliser la session entraîne cette erreur, plutôt que l'erreur initiale`InvalidSessionException`.

Si vous rencontrez cette exception, nous vous recommandons de créer une nouvelle session et de réessayer la transaction. Nous recommandons également d'utiliser la dernière version du pilote QLDB, qui gère le pool de sessions et son état au nom de l'application.

RateExceededException

Message : Le taux a été dépassé

QLDB a limité un client en fonction de l'identité de l'appelant. QLDB applique la limitation par région et par compte à l'aide d'un algorithme de limitation des compartiments à jetons. QLDB le fait pour améliorer la performance du service et pour garantir une utilisation équitable pour tous les clients QLDB. Par exemple, le fait d'essayer d'acquérir un grand nombre de sessions simultanées à l'aide de cette`StartSessionRequest` opération peut entraîner un ralentissement.

Pour préserver la santé de votre application et atténuer toute nouvelle limitation, vous pouvez réessayer cette exception en utilisant Exponential Backoff et Jitter. Le concept de recul exponentiel consiste à utiliser des temps d'attente progressivement plus longs entre les nouvelles tentatives pour des réponses d'erreur consécutives. Nous vous recommandons d'utiliser la dernière version du pilote QLDB. Le pilote applique une politique de relance par défaut qui utilise un recul et une instabilité exponentiels pour réessayer automatiquement en cas d'exceptions telles que celle-ci.

La dernière version du pilote QLDB peut également vous aider si votre application est constamment limitée par QLDB pour les`StartSessionRequest` appels. Le chauffeur gère un pool de sessions qui sont réutilisées pour toutes les transactions, ce qui peut contribuer à réduire le nombre d'`StartSessionRequest`appels passés par votre application. Pour demander une augmentation des limitations d'API, veuillez contacter le AWS SupportCentre.

LimitExceededException

Message : dépassement de la limite de session

Un registre a dépassé son quota (également appelé limite) en termes de nombre de sessions actives. Ce quota est défini dansQuotas et limites d'Amazon QLDB. Le nombre de sessions actives d'un registre finit par être constant, et les livres dont le quota est constamment proche du quota peuvent régulièrement faire l'objet de cette exception.

Pour préserver l'intégrité de votre application, nous vous recommandons de réessayer cette exception. Pour éviter cette exception, assurez-vous de ne pas avoir configuré plus de 1 500 sessions simultanées à utiliser pour un seul registre pour tous les clients. Par exemple, vous pouvez utiliser la maxConcurrentTransactionsméthode du pilote Amazon QLDB pour Java afin de configurer le nombre maximum de sessions disponibles dans une instance de pilote.

QldbClientException

Message : un résultat diffusé n'est valide que lorsque la transaction parent est ouverte

La transaction est clôturée et ne peut pas être utilisée pour récupérer les résultats depuis QLDB. Une transaction est clôturée lorsqu'elle est validée ou annulée.

Cette exception se produit lorsque le client travaille directement avec l'`Transaction`objet et essaie de récupérer les résultats de QLDB après avoir validé ou annulé une transaction. Pour atténuer ce problème, le client doit lire les données avant de clôturer la transaction.

Exportation des données de journal

Cette section répertorie les exceptions courantes que QLDB peut renvoyer lorsque vous exportez des données de journal depuis un registre vers un compartiment Amazon S3. Pour en savoir plus sur cette fonction, consultez Exportation de données de journal depuis Amazon QLDB.

Chaque exception inclut le message d'erreur spécifique, suivi d'une brève description et de suggestions de solutions possibles.

AccessDeniedException

Message : Utilisateur : *UserArn n'est pas autorisé à effectuer : iam :PassRole sur une ressource : roleARN*

Vous n'êtes pas autorisé à transférer un rôle IAM au service QLDB. QLDB nécessite un rôle pour toutes les demandes d'exportation de journaux, et vous devez être autorisé à transmettre ce rôle à QLDB. Le rôle fournit à QLDB des autorisations d'écriture dans le compartiment Amazon S3 que vous avez spécifié.

Vérifiez que vous définissez une politique IAM qui autorise l'exécution de l'opération d'`PassRole`API sur la ressource de rôle IAM que vous avez spécifiée pour le service QLDB (`qldb.amazonaws.com`). Pour un exemple de stratégie, consultez Exemples de politiques basées sur l'identité pour Amazon QLDB.

IllegalArgumentException

Message : QLDB a rencontré une erreur lors de la validation de la configuration S3 : *errorCode ErrorMessage*

Cette erreur peut être parce que le compartiment Amazon S3 fourni n'existe pas dans Amazon S3. Ou bien, QLDB ne dispose pas de suffisamment d'autorisations pour écrire des objets dans le compartiment Amazon S3 que vous avez spécifié.

Vérifiez que le nom du compartiment S3 que vous avez indiqué dans votre demande de tâche d'exportation est correct. Pour plus d'informations sur l'attribution de noms aux compartiments, consultez la section Restrictions et limites des compartiments du Guide de l'utilisateur Amazon Simple Storage Service.

Vérifiez également que vous définissez une politique pour le compartiment que vous avez spécifié qui accorde`PutObject` des`PutObjectAcl` autorisations au service QLDB (`qldb.amazonaws.com`). Pour en savoir plus, veuillez consulter la section Autorisations d'exportation.

IllegalArgumentException

Message : réponse inattendue d'Amazon S3 lors de la validation de la configuration S3. Réponse de S3 : *errorCode errorMessage*

La tentative d'écriture des données d'exportation du journal dans le compartiment S3 fourni a échoué avec la réponse d'erreur Amazon S3 fournie. Pour plus d'informations sur les causes

possibles, consultez la section Résolutions des problèmes liés à Amazon S3 dans le Guide de l'utilisateur Amazon Simple Storage Service.

IllegalArgumentException

Message : le préfixe du compartiment Amazon S3 ne doit pas dépasser 128 caractères

Le préfixe fourni dans la demande d'exportation du journal contient plus de 128 caractères.

IllegalArgumentException

Message : La date de début ne doit pas être supérieure à la date de fin

Les deux`InclusiveStartTime` et`ExclusiveEndTime` doivent être au format date et heure ISO 8601 et en temps universel coordonné (UTC).

IllegalArgumentException

Message : La date de fin ne peut pas être dans le future

Les deux`InclusiveStartTime``ExclusiveEndTime` doivent être au format`ISO 8601` date et heure et en UTC.

IllegalArgumentException

Message : Le paramètre de chiffrement des objets (S3EncryptionConfiguration) fourni n'est pas compatible avec une cléAWS Key Management Service (AWS KMS)

Vous avez fourni un`KMSKeyArn``ObjectEncryptionType` de l'un`NO_ENCRYPTION` ou l'autre`SSE_S3`. Vous ne pouvez fournir un client géré queAWS KMS key pour un type de cryptage d'objets de`SSE_KMS`. Pour plus d'informations sur les options de chiffrement côté serveur dans Amazon S3, consultez Protection des données à l'aide d'un chiffrement côté serveur dans le Guide du développeur Amazon S3.

LimitExceededException

Message : dépassement de la limite de 2 tâches d'exportation de journaux exécutées simultanément

QLDB applique une limite par défaut de deux tâches d'exportation de journaux simultanées.

Diffusion des données du journal

Cette section répertorie les exceptions courantes que QLDB peut renvoyer lorsque vous diffusez des données de journal depuis un registre vers Amazon Kinesis Data Streams. Pour en savoir plus sur cette fonction, consultez Diffusion en continu de données de journaux depuis Amazon QLDB.

Chaque exception inclut le message d'erreur spécifique, suivi d'une brève description et de suggestions de solutions possibles.

AccessDeniedException

Message : Utilisateur : *`UserArn n'est pas autorisé à effectuer : iam :PassRole sur une ressource : roleARN`*

Vous n'êtes pas autorisé à transférer un rôle IAM au service QLDB. QLDB nécessite un rôle pour toutes les demandes de flux de journal, et vous devez être autorisé à transmettre ce rôle à QLDB. Le rôle fournit à QLDB des autorisations d'écriture dans la ressource Amazon Kinesis Data Streams que vous avez spécifiée.

Vérifiez que vous définissez une politique IAM qui autorise l'exécution de l'opération d'`PassRole`API sur la ressource de rôle IAM que vous avez spécifiée pour le service QLDB (`qldb.amazonaws.com`). Pour un exemple de stratégie, consultez Exemples de politiques basées sur l'identité pour Amazon QLDB.

IllegalArgumentException

Message : QLDB a rencontré une erreur lors de la validation de Kinesis Data Streams : Réponse de Kinesis : *`errorCode errorMessage`*

Cette erreur peut être due au fait que la ressource Kinesis Data Streams fournie n'existe pas. Ou bien, QLDB ne dispose pas des autorisations suffisantes pour écrire des enregistrements de données dans le flux de données Kinesis que vous avez spécifié.

Vérifiez que le flux de données Kinesis que vous fournissez dans votre demande de flux est correct. Pour plus d'informations, consultez la section Création et mise à jour de flux de données dans le Guide du développeur Amazon Kinesis Data Streams.

Vérifiez également que vous définissez une politique pour le flux de données Kinesis que vous avez spécifié qui accorde au service QLDB (`qldb.amazonaws.com`) des autorisations pour les actions suivantes. Pour plus d'informations, consultez Autorisations de diffusion.

- `kinesis:PutRecord`
- `kinesis:PutRecords`
- `kinesis:DescribeStream`
- `kinesis:ListShards`

IllegalArgumentException

Message : réponse inattendue de Kinesis Data Streams lors de la validation de la configuration Kinesis. Réponse de Kinesis : *`errorCode errorMessage`*

La tentative d'écriture d'enregistrements de données dans le flux de données Kinesis fourni a échoué avec la réponse d'erreur Kinesis fournie. Pour plus d'informations sur les causes possibles, consultez la section Résolutions des problèmes liés aux producteurs Amazon Kinesis Data Streams dans le Guide du développeur Amazon Kinesis Data Streams.

IllegalArgumentException

Message : La date de début ne doit pas être supérieure à la date de fin.

Les deux`InclusiveStartTime` et`ExclusiveEndTime` doivent être au format date et heure ISO 8601 et en temps universel coordonné (UTC).

IllegalArgumentException

Message : La date de début ne peut pas être dans le future.

Les deux`InclusiveStartTime``ExclusiveEndTime` doivent être au format`ISO 8601` date et heure et en UTC.

LimitExceededException

Message : dépassement de la limite de 5 flux de journaux exécutés simultanément vers Kinesis Data Streams

QLDB applique une limite par défaut de cinq flux de journaux simultanés.

Vérification des données du journal

Cette section répertorie les exceptions courantes que QLDB peut renvoyer lorsque vous vérifiez des données de journal dans un registre. Pour en savoir plus sur cette fonction, consultez Vérification des données dans Amazon QLDB.

Chaque exception inclut le message d'erreur spécifique, suivi des opérations d'API qui peuvent le générer, d'une brève description et de suggestions de solutions possibles.

IllegalArgumentException

Message : La valeur Ion fournie n'est pas valide et ne peut pas être analysée.

Opérations d'API :`GetDigest, GetBlock, GetRevision`

Assurez-vous de fournir une valeur Amazon Ion valide avant de réessayer votre demande.

IllegalArgumentException

Message : L'adresse de bloc fournie n'est pas valide.

Opérations d'API :`GetDigest, GetBlock, GetRevision`

Assurez-vous de fournir une adresse de blocage valide avant de réessayer votre demande. Une adresse de bloc est une structure Amazon Ion qui comporte deux champs :`strandId` et`sequenceNo`.

Par exemple : `{strandId:"BlFTjlSXze9BIh1K0szcE3",sequenceNo:14}`

IllegalArgumentException

Message : Le numéro de séquence de l'adresse de l'astuce de synthèse fournie est supérieur au dernier enregistrement enregistré du volet.

Opérations d'API :`GetDigest, GetBlock, GetRevision`

L'adresse du conseil de synthèse que vous fournissez doit avoir un numéro de séquence inférieur ou égal au numéro de séquence du dernier enregistrement validé du volet de journal. Avant de réessayer votre demande, assurez-vous de fournir une adresse de conseil contenant un numéro de séquence valide.

IllegalArgumentException

Message : L'ID de chaîne de l'adresse de bloc fournie n'est pas valide.

Opérations d'API :`GetDigest, GetBlock, GetRevision`

L'adresse de bloc que vous fournissez doit avoir un identifiant de volet qui correspond à l'identifiant de volet du journal. Avant de réessayer votre demande, assurez-vous de fournir une adresse de bloc avec un identifiant de volet valide.

IllegalArgumentException

Message : Le numéro de séquence de l'adresse de bloc fournie est supérieur au dernier enregistrement validé du brin.

Opérations d'API :`GetBlock, GetRevision`

L'adresse de bloc que vous fournissez doit avoir un numéro de séquence inférieur ou égal au numéro de séquence du dernier enregistrement validé du brin. Avant de réessayer votre demande, assurez-vous de fournir une adresse de bloc avec un numéro de séquence valide.

IllegalArgumentException

Message : L'ID de brin de l'adresse de bloc fournie doit correspondre à l'ID de brin de l'adresse du condensé fournie.

Opérations d'API :`GetBlock, GetRevision`

Vous ne pouvez vérifier la révision ou le blocage d'un document que s'il figure dans le même volet de journal que le résumé que vous fournissez.

IllegalArgumentException

Message : Le numéro de séquence de l'adresse de bloc fournie ne doit pas être supérieur au numéro de séquence de l'adresse du résumé fourni.

Opérations d'API :`GetBlock, GetRevision`

Vous ne pouvez vérifier la révision ou le blocage d'un document que s'il est couvert par le résumé que vous fournissez. Cela signifie qu'il a été publié dans le journal avant l'adresse du résumé.

IllegalArgumentException

Message : L'ID de document fourni n'a pas été trouvé dans le bloc à l'adresse de bloc spécifiée.

Fonctionnement de l'API :`GetRevision`

L'ID du document que vous fournissez doit figurer dans l'adresse de bloc que vous avez fournie. Avant de réessayer votre demande, assurez-vous que ces deux paramètres sont cohérents.

Référence Amazon QLDB

Amazon QLDB prend en charge un sous-ensemble du langage de requête PartiQL. Les rubriques suivantes décrivent l'implémentation QLDB de PartiQL.

> **Note**
>
> - QLDB ne prend pas en charge toutes les opérations PartiQLDB
> - Toutes les instructions PartiQL de QLDB sont soumises à des limites de transactions, telles que définies dansQuotas et limites d'Amazon QLDB.
> - Cette référence fournit des exemples de syntaxe et d'utilisation de base des instructions PartiQL que vous exécutez manuellement sur la console QLDB ou dans l'interpréteur QLDB. Pour des exemples de code montrant comment exécuter par programmation des instructions similaires à l'aide du pilote QLDB, consultez les didacticiels dansDémarrage.

Rubriques

Qu'est-ce que PartiQL ?

Le langage PartiQL fournit un accès aux requêtes compatible SQL sur plusieurs magasins de données contenant des données structurées, des données semi-structurées et des données imbriquées. Il est largement utilisé sur Amazon et est désormais disponible dans de nombreuses applicationsServices AWS, dont QLDB.

Pour la spécification de PartiQL et un didacticiel sur le langage de requête de base, consultez la Documentation PartiQL.

PartiQL étend SQL-92 pour prendre en charge les documents au format de données Amazon Ion. Pour obtenir des informations sur Amazon IonRéférence du format de données Amazon Ion dans Amazon QLDB

PartiQL

Pour exécuter des requêtes PartiQL dans QLDB, vous pouvez utiliser l'une des méthodes suivantes :

- L'éditeur PartiQL sur leAWS Management Console for QLDB
- Le shell QLDB en ligne de commande
- Un pilote QLDBAWS fourni pour exécuter des requêtes par programmation

Pour plus d'informations sur l'utilisation de ces méthodes pour accéder à QLDB, veuillez consulterAccès à Amazon QLDB.

Pour savoir comment contrôler l'accès afin d'exécuter chaque commande PartiQL sur des tables spécifiques, consultezCommencer à utiliser le mode d'autorisation standard dans Amazon QLDB.

Astuces rapides pour PartiQL dans QLDB

Vous trouverez ci-dessous un bref résumé des conseils et des meilleures pratiques pour utiliser PartiQL dans QLDB :

- Comprenez les limites de simultanéité et de transaction : toutes les déclarations, y compris les`SELECT` requêtes, sont soumises à des conflits et à des limites de transaction optimistes (OCC), y compris un délai d'expiration de 30 secondes pour les transactions.

- Utilisez des index : utilisez des index à cardinalité élevée et exécutez des requêtes ciblées pour optimiser vos déclarations et éviter de scanner des tableaux complets. Pour en savoir plus, consultez Optimisation des performances des données.
- Utiliser des prédicats d'égalité : les recherches indexées nécessitent un opérateur d'égalité (`=`ou`IN`). Les opérateurs d'inégalité (`<>`,`LIKE`,,`BETWEEN`) ne sont pas éligibles aux recherches indexées et donnent lieu à des analyses de tableaux complets.
- Utiliser uniquement les jointures internes : QLDB ne prend en charge que les jointures internes. Il est recommandé de joindre des champs indexés pour chaque table que vous joignez. Choisissez des indices de cardinalité élevée pour les critères de jointure et les prédicats d'égalité.

Conventions de référence Amazon QLDB PartiQL

Cette section explique les conventions utilisées pour écrire la syntaxe des commandes, fonctions et expressions PartiQL décrites dans la référence Amazon QLDB PartiQL. Ces conventions ne doivent pas être confondues avec la syntaxe et la sémantique du langage de requête PartiQL lui-même.

Caractère	Description
CAPS	Les mots en lettres majuscules sont des mots clés.
[ ]	Les crochets indiquent des arguments ou des clauses facultatifs. Plusieurs arguments entre crochets signifient que vous pouvez choisir n'importe quel nombre d'arguments. En outre, les arguments entre crochets placés sur des lignes séparées indiquent que l'analyseur QLDB s'attend à ce que les arguments soient dans l'ordre où ils apparaissent dans la syntaxe.
|	Les barres verticales indiquent que vous pouvez choisir entre les arguments.
italique rouge	Les mots en italique rouge correspondent à des espaces réservés. Insérez la valeur appropriée à la place du mot en italique rouge.
. . .	Les trois points de suspension indiquent que vous pouvez répéter l'élément précédent.
'	Les valeurs entre apostrophes droites signifient que vous devez entrer les apostrophes. Dans PartiQL, les guillemets simples désignent des valeurs de chaîne ou des noms de champs dans les structures Amazon Ion.

Caractère	Description
"	Les valeurs entre apostrophes droites signifient que vous devez entrer les apostrophes. Dans PartiQL, les guillemets doubles désignent les identific ateurs entre guillemets.
`	Les valeurs indiquées dans les marqueurs inverses indiquent que vous devez saisir les marqueurs inverses. Dans PartiQL, les crochets inverses indiquent les valeurs littérales d'Ion.

Types de données dans Amazon QLDB

Amazon QLDB stocke les documents au format Amazon Ion. Amazon Ion est un format de sérialisation de données (à la fois sous forme de texte et sous forme de code binaire) qui est un sur-ensemble de JSON. Le tableau suivant répertorie les types de données Ion que vous pouvez utiliser dans les documents QLDB.

Type de données	Description
`null`	Une valeur nulle générique
`bool`	Valeurs booléennes
`int`	Entiers signés de taille arbitraire
`decimal`	Nombres réels codés en décimales de précision arbitraire
`float`	Numéros à virgule flottante codés en binaire (IEEE 64 bits)
`timestamp`	Moments de date/heure/fuseau horaire d'une précision arbitraire
`string`	Littéraux de texte Unicode
`symbol`	Atomes symboliques Unicode (identificateurs)

Type de données	Description
`blob`	Données binaires encodées par l'utilisateur
`clob`	Données de texte dont le codage est défini par l'utilisateur
`struct`	Collections non ordonnées de paires nom-valeur
`list`	Collections de valeurs hétérogènes ordonnées

Consultez le document de spécifications Ion sur le GitHub site Amazon pour obtenir la liste complète des types de données Ion Core, avec des descriptions complètes et des détails de mise en forme des valeurs.

Documents Amazon QLDB

Amazon QLDB stocke les enregistrements de données sous forme de documents, qui ne sont que`struct` des objets Amazon Ion insérés dans un tableau. Pour la spécification Ion, consultez le GitHub site Amazon Ion.

Rubriques

- Structure d'Ion
- Cartographie des types d'ions partiels
- Numéro du document

Structure d'Ion

Comme JSON, les documents QLDB sont composés de paires nom-valeur dans la structure suivante.

```
{
  name1: value1,
  name2: value2,
  name3: value3,
  ...
```

```
  nameN: valueN
}
```

Les noms sont des symboles et les valeurs ne sont pas limitées. Chaque paire nom-valeur est appelée champ. La valeur d'un champ peut être n'importe quel ionTypes de données, y compris les types de conteneurs : structures imbriquées, listes et listes de structures.

Tout comme JSON, `astruct` est indiqué par des accolades (`{...}`) et `alist` est indiqué par des crochets (`[...]`). L'exemple suivant est un document issu des exemples de donnéesMise en route avec la console Amazon QLDB qui contient des valeurs de différents types.

```
{
    VIN: "1N4AL11D75C109151",
    LicensePlateNumber: "LEWISR261LL",
    State: "WA",
    City: "Seattle",
    PendingPenaltyTicketAmount: 90.25,
    ValidFrom: 2017-08-21T,
    ValidTo: 2020-05-11T,
    Owners: {
        PrimaryOwner: { PersonId: "294jJ3YUoH1IEEm8GSabOs" },
        SecondaryOwners: [{ PersonId: "5Ufgdlnj06gF5CWcOIu64s" }]
    }
}
```

> **Important**
>
> Dans Ion, les guillemets doubles indiquent des valeurs de chaîne et les symboles sans guillemets représentent des noms de champs. Mais dans PartiQL, les guillemets simples désignent à la fois les chaînes et les noms de champs.
> Cette différence de syntaxe permet au langage de requête PartiQL de maintenir la compatibilité SQL et au format de données Amazon Ion de maintenir la compatibilité JSON. Pour plus de détails sur la syntaxe et la sémantique de PartiQL dans QLDB, consultezInterroger Ion avec PartiQL.

Cartographie des types d'ions partiels

Dans QLDB, PartiQL étend le système de types SQL pour couvrir le modèle de données Ion. Ce mappage est décrit comme suit :

- Les types scalaires SQL sont couverts par leurs homologues Ion. Par exemple :
  - `CHAR`et`VARCHAR` sont des séquences Unicode correspondant au`string` type Ion.
  - `NUMBER`correspond au`decimal` type Ion.
- L`struct` type d'Ion est équivalent à un tuple SQL, qui représente traditionnellement une ligne de table.
  - Toutefois, avec un contenu ouvert et sans schéma, les requêtes qui s'appuient sur la nature ordonnée d'un tuple SQL ne sont pas prises en charge (comme l'ordre de sortie de`SELECT *`).
- En plus de`NULL`, PartiQL possède un`MISSING` type. Il s'agit d'une spécialisation`NULL` et indique l'absence de domaine. Ce type est nécessaire car`struct` les champs ioniques peuvent être clairsemés.

Numéro du document

QLDB attribue un identifiant de document à chaque document que vous insérez dans un tableau. Tous les identifiants attribués par le système sont des identifiants uniques universels (UUID) qui sont chacun représentés dans une chaîne codée en Base62 (par exemple,`3Qv67yjXEwB9SjmvkuG6Cp`). Pour plus d'informations, veuillez consulter Identifiants uniques dans Amazon QLDB.

Chaque révision de document est identifiée de manière unique par une combinaison de l'identifiant du document et d'un numéro de version en base zéro.

Les champs d'ID et de version du document sont inclus dans les métadonnées du document, que vous pouvez consulter dans la vue validée (la vue définie par le système d'une table). Pour plus d'informations sur les vues dans QLDBConcepts de base Pour en savoir plus sur les métadonnées, veuillez consulterInterroger les métadonnées d'un document.

Interroger Ion avec PartiQL dans Amazon QLDB

Lorsque vous interrogez des données dans Amazon QLDB, vous rédigez des instructions au format PartiQL, mais QLDB renvoie les résultats au format Amazon Ion. PartiQL est censé être compatible avec SQL, tandis que Ion est une extension de JSON. Cela entraîne des différences syntaxiques entre la façon dont vous notez les données dans vos instructions de requête et la façon dont les résultats de votre requête sont affichés.

Cette section décrit la syntaxe et la sémantique de base permettant d'exécuter des instructions PartiQL manuellement à l'aide de la console QLDB ou du shell QLDB.

> **Tip**
>
> Lorsque vous exécutez des requêtes PartiQL par programmation, la meilleure pratique consiste à utiliser des instructions paramétrées. Vous pouvez utiliser un point d'interrogation (?) comme espace réservé à une variable de liaison dans vos déclarations pour éviter ces règles de syntaxe. C'est également plus sûr et plus efficace.
> Pour en savoir plus, consultez les didacticiels suivants dans la section Prise en main du pilote :
>
> - Java :Dacticiel de démarrage rapide |Référence du livre de recettes
> - .NET :Ddémarrage rapide |Référence de livre de cuisine
> - Allez :Didacticiel de démarrage rapide |Référence de livre de recettes
> - Node.js :Didacticiel de démarrage rapide |Référence de livre de cuisine
> - Python :Didacticiel de démarrage rapide |Référence de livre de recettes

Rubriques

Syntaxe et sémantique

Lorsque vous utilisez la console QLDB ou le shell QLDB pour interroger des données Ion, voici la syntaxe et la sémantique fondamentales de PartiQL :

Sensibilité à la casse

Tous les noms d'objets du système QLDB, y compris les noms de champs, les noms de table et les noms de registre, distinguent les majuscules et minuscules.

Valeurs de chaîne

Dans Ion, les guillemets doubles (`"..."`) indiquent une chaîne.

Dans PartiQL, les guillemets simples (`'...'`) indiquent une chaîne.

Symboles et identificateurs

Dans Ion, les guillemets simples (`'...'`) indiquent un symbole. Un sous-ensemble de symboles dans Ion appelés identificateurs est représenté par du texte sans guillemets.

Dans PartiQL, les guillemets doubles (`"..."`) indiquent un identifiant PartiQL entre guillemets, tel qu'un mot réservé utilisé comme nom de table. Le texte sans guillemets représente un identifiant PartiQL normal, tel qu'un nom de table qui n'est pas un mot réservé.

Ion

Tous les littéraux Ion peuvent être désignés par des backticks (`` `...` ``) dans une instruction PartiQL.

Noms de champ

Les noms des champs Ion sont sensibles à la casse. PartiQL vous permet de désigner les noms de champs par des guillemets simples dans une instruction DML. Il s'agit d'une alternative abrégée à l'utilisation de la`cast` fonction PartiQL pour définir un symbole. C'est également plus intuitif que d'utiliser des marqueurs inverses pour désigner un symbole ionique littéral.

Littéraux

Les littéraux du langage de requête PartiQL correspondent aux types de données Ion, comme suit :

Scalaires

Respectez la syntaxe SQL, le cas échéant, comme décrit dansCartographie des types d'ions partiels la section. Par exemple :

- `5`
- `'foo'`
- `null`

Structures

Également appelés tuples ou objets dans de nombreux formats et autres modèles de données.

Désigné par des accolades (`{...}`) avec des`struct` éléments séparés par des virgules.

- `{ 'id' : 3, 'arr': [1, 2] }`

Listes

Également désignées sous le nom de tableaux.

Indiqué par des crochets (`[...]`) avec des éléments de liste séparés par des virgules.

- `[ 1, 'foo' ]`

Sacs

Collections non ordonnées dans PartiQL.

Désigné par des crochets à double angle (`<<...>>`) avec les éléments du sac séparés par des virgules. Dans QLDB Toutefois, un sac ne peut pas être imbriqué dans les documents d'un tableau.

- `<< 1, 'foo' >>`

Exemple

Voici un exemple de syntaxe pour une`INSERT` instruction comportant différents types d'ions.

```
INSERT INTO VehicleRegistration VALUE
{
    'VIN' : 'KM8SRDHF6EU074761', --string
    'RegNum' : 1722, --integer
    'State' : 'WA',
    'City' : 'Kent',
    'PendingPenaltyTicketAmount' : 130.75, --decimal
    'Owners' : { --nested struct
        'PrimaryOwner' : { 'PersonId': '294jJ3YUoH1IEEm8GSabOs' },
        'SecondaryOwners' : [ --list of structs
            { 'PersonId' : '1nmeDdLo3AhGswBtyM1eYh' },
            { 'PersonId': 'IN7MvYtUjkp1GMZu0F6CG9' }
        ]
    },
    'ValidFromDate' : `2017-09-14T`, --Ion timestamp literal with day precision
    'ValidToDate' : `2020-06-25T`
}
```

Notation en forme de coche

PartiQL couvre entièrement tous les types de données Ion, ce qui vous permet d'écrire n'importe quelle instruction sans utiliser de backticks. Mais dans certains cas, cette syntaxe littérale Ion peut rendre vos déclarations plus claires et plus concises.

Par exemple, pour insérer un document contenant des valeurs d'horodatage et de symbole Ion, vous pouvez écrire l'instruction suivante en utilisant uniquement la syntaxe PartiQL.

```
INSERT INTO myTable VALUE
{
    'myTimestamp': to_timestamp('2019-09-04T'),
    'mySymbol': cast('foo' as symbol)
}
```

C'est assez détaillé, vous pouvez donc utiliser des backticks pour simplifier votre déclaration.

```
INSERT INTO myTable VALUE
{
    'myTimestamp': `2019-09-04T`,
    'mySymbol': `foo`
}
```

Vous pouvez également placer l'ensemble de la structure dans des barres inverses pour enregistrer quelques touches supplémentaires.

```
INSERT INTO myTable VALUE
`{
    myTimestamp: 2019-09-04T,
    mySymbol: foo
}`
```

Important

Les chaînes et les symboles sont des classes différentes dans PartiQL. Cela signifie que même s'ils ont le même texte, ils ne sont pas égaux. Par exemple, les expressions PartiQL suivantes sont évaluées sur des valeurs Ion différentes.

```
'foo'
```

```
`foo`
```

Navigation par chemin

Lorsque vous rédigez un langage de manipulation de données (DML) ou des instructions de requête, vous pouvez accéder aux champs des structures imbriquées à l'aide d'étapes de chemin. PartiQL prend en charge la notation par points pour accéder aux noms de champs d'une structure parent. L'exemple suivant accède au`Model` champ d'un parent`Vehicle`.

```
Vehicle.Model
```

Pour accéder à un élément spécifique d'une liste, vous pouvez utiliser l'opérateur entre crochets pour indiquer un nombre ordinal à base zéro. L'exemple suivant accède à l'élément de`SecondaryOwners` avec un nombre ordinal de2. En d'autres termes, il s'agit du troisième élément de la liste.

```
SecondaryOwners[2]
```

Aliasing

QLDB prend en charge le contenu et les schémas ouverts. Ainsi, lorsque vous accédez à des champs spécifiques d'une instruction, le meilleur moyen de vous assurer d'obtenir les résultats escomptés est d'utiliser des alias. Par exemple, si vous ne spécifiez pas d'alias explicite, le système en génère un implicite pour vos`FROM` sources.

```
SELECT VIN FROM Vehicle
--is rewritten to
SELECT Vehicle.VIN FROM Vehicle AS Vehicle
```

Mais les résultats sont imprévisibles en cas de conflits de noms de champs. Si un autre champ nommé`VIN` existe dans une structure imbriquée au sein des documents, les`VIN` valeurs renvoyées par cette requête peuvent vous surprendre. La bonne pratique consiste à écrire plutôt la déclaration suivante. Cette requête est déclarée`v` comme un alias qui s'étend sur la`Vehicle` table. Le`AS` mot-clé est facultatif.

```
SELECT v.VIN FROM Vehicle [ AS ] v
```

L'alias est particulièrement utile pour accéder à des collections imbriquées au sein d'un document. Par exemple, l'instruction suivante déclareo qu'il s'agit d'un alias qui couvre la collection`VehicleRegistration.Owners`.

```
SELECT o.SecondaryOwners
FROM VehicleRegistration AS r, @r.Owners AS o
```

Le@ personnage est ici techniquement facultatif. Mais cela indique explicitement que vous souhaitez que la`Owners` structure soit intégrée`VehicleRegistration`, et non qu'une collection différente soit nommée`Owners` (s'il en existe une).

Spécification PartiQL

Pour de plus amples informations sur le langage de requête PartiQL, veuillez consulter la spécification PartiQL.

Commandes partiQL dans Amazon QLDB

PartiQL étend SQL-92 pour prendre en charge les documents au format de données Amazon Ion. Amazon QLDB

Pour savoir comment contrôler l'accès afin d'exécuter chaque commande PartiQL sur des tables spécifiques, consultezCommencer à utiliser le mode d'autorisation standard dans Amazon QLDB.

> **Note**
>
> - QLDB ne prend pas en charge toutes les commandes PartiQLDB
> - Toutes les instructions PartiQL de QLDB sont soumises à des limites de transactions, telles que définies dansQuotas et limites d'Amazon QLDB.
> - Cette référence fournit des exemples de syntaxe et d'utilisation de base des instructions PartiQL que vous exécutez manuellement sur la console QLDB ou dans l'interpréteur QLDB. Pour des exemples de code montrant comment exécuter des instructions similaires à l'aide d'un langage de programmation compatible, consultez les didacticiels dansDémarrage.

Déclarations DDL (langage de définition des données)

Le langage de définition des données (Data Definition language, DDL) est l'ensemble d'instructions PartiQL que vous utilisez pour gérer les objets de base de données, tels que les tables et les index. Vous utilisez le protocole DDL pour créer et supprimer ces objets.

- CREATE INDEX
- CREATE TABLE
- DROP INDEX
- DROP TABLE
- TABLEAU DE DÉBALLAGE

Déclarations DML (langage de manipulation de données)

Le langage de manipulation des données (Data manipulation language, DML) est l'ensemble d'instructions PartiQL que vous utilisez pour gérer les données dans des tables QLDB. Vous utilisez des instructions DML pour ajouter, modifier ou supprimer des données dans une table.

Les instructions DML et de langage de requête prises en charge sont les suivantes :

- DELETE
- DE (INSÉRER, SUPPRIMER ou DÉFINIR)
- INSERT
- SELECT
- MISE A JOUR

Commande CREATE INDEX dans Amazon QLDB

Dans Amazon QLDB, utilisez la`CREATE INDEX` commande pour créer un index pour un champ de document dans une table.

Pour savoir comment contrôler l'accès afin d'exécuter cette commande PartiQL sur des tables spécifiques, consultezCommencer à utiliser le mode d'autorisation standard dans Amazon QLDB.

> **Important**
>
> QLDB a besoin d'un index pour rechercher efficacement un document. Sans index, QLDB doit effectuer une analyse complète de la table lors de la lecture de documents. Cela peut entraîner des problèmes de performances sur des tables de grande taille, notamment des conflits de simultanéité et des délais de transaction.
> Pour éviter de scanner des tables, vous devez exécuter des instructions avec une clause de`WHERE` prédicat à l'aide d'un opérateur d'égalité (`=`ou`IN`) sur un champ indexé ou un identifiant de document. Pour plus d'informations, veuillez consulter Optimisation des performances des données.

Tenez compte des contraintes suivantes lors de la création d'index :

- Un index ne peut être créé que sur un seul champ de niveau supérieur. Les index composites, imbriqués, uniques et basés sur des fonctions ne sont pas pris en charge.
- Vous pouvez créer un index sur tous les types de données Ion, y compris`list` et`struct`. Toutefois, vous ne pouvez effectuer la recherche indexée que par égalité de la valeur totale des ions, quel que soit le type d'ion. Par exemple, lorsque vous utilisez un`list` type comme index, vous ne pouvez pas effectuer de recherche indexée par un élément de la liste.
- Les performances des requêtes ne sont améliorées que lorsque vous utilisez un prédicat d'égalité ; par exemple,`WHERE indexedField = 123` ou`WHERE indexedField IN (456, 789)`.

  QLDB ne respecte pas les inégalités dans les prédicats de requête. Par conséquent, les analyses filtrées par plage de valeurs ne sont pas mises en œuvre.
- Les noms des champs indexés sont sensibles à la casse et peuvent contenir un maximum de 128 caractères.
- La création d'index dans QLDB est asynchrone. Le temps nécessaire à la création d'un index sur une table non vide varie en fonction de la taille de la table. Pour plus d'informations, veuillez consulter Gestion des index.

Rubriques

- Syntaxe
- Paramètres
- Valeur renvoyée

- Exemples
- Exécution par programmation à l'aide du pilote

Syntaxe

```
CREATE INDEX ON table_name (field)
```

Paramètres

table_name

Nom de la table dans laquelle vous souhaitez créer l'index. La table doit déjà exister.

Le nom de la table est sensible à la casse.

champ

Nom du champ du document pour lequel créer l'index. Le champ doit être un attribut de niveau supérieur.

Les noms des champs indexés sont sensibles à la casse et peuvent contenir un maximum de 128 caractères.

Vous pouvez créer un index sur tous les types de données Amazon Ion, y compris`list` et`struct`. Toutefois, vous ne pouvez effectuer la recherche indexée que par égalité de la valeur totale des ions, quel que soit le type d'ion. Par exemple, lorsque vous utilisez un`list` type comme index, vous ne pouvez pas effectuer de recherche indexée par un élément de la liste.

Valeur renvoyée

`tableId`— L'ID unique de la table sur laquelle vous avez créé l'index.

Exemples

```
CREATE INDEX ON VehicleRegistration (LicensePlateNumber)
```

```
CREATE INDEX ON Vehicle (VIN)
```

Exécution par programmation à l'aide du pilote

Pour savoir comment exécuter cette instruction par programmation à l'aide du pilote QLDB, consultez les didacticiels suivants dans la section Prise en main du pilote :

- Java :Dacticiel de démarrage rapide |Référence du livre de recettes
- .NET :Ddémarrage rapide |Référence de livre de cuisine
- Allez :Didacticiel de démarrage rapide |Référence de livre de recettes
- Node.js :Didacticiel de démarrage rapide |Référence de livre de cuisine
- Python :Didacticiel de démarrage rapide |Référence de livre de recettes

Commande CREATE TABLE dans Amazon QLDB

Dans Amazon QLDB, utilisez la`CREATE TABLE` commande pour créer une nouvelle table.

Les tables ont des noms simples sans espaces de noms. QLDB prend en charge le contenu ouvert et n'applique pas de schéma. Vous ne définissez donc pas d'attributs ou de types de données lors de la création de tables.

> **Note**
>
> Pour savoir comment contrôler l'accès afin d'exécuter cette commande PartiQL dans un registre, consultezCommencer à utiliser le mode d'autorisation standard dans Amazon QLDB.

Rubriques

Syntaxe

```
CREATE TABLE table_name [ WITH (aws_tags = `{'key': 'value'}`) ]
```

Paramètres

table_name

Nom unique de la table à créer. Une table active portant le même nom ne doit pas déjà exister. Les contraintes de dénomination sont les suivantes :

- Doit contenir entre 1 et 128 caractères alphanumériques ou traits de soulignement.
- Doit comporter une lettre ou un trait de soulignement pour le premier caractère.
- Peut comporter n'importe quelle combinaison de caractères alphanumériques et de traits de soulignement pour les autres caractères.
- respecte la casse.
- Il ne doit pas s'agir d'un mot réservé QLDB PartiQL.

« ***clé*** » : « ***valeur*** »

(Facultatif) Balises à associer à la ressource de table lors de la création. Chaque balise est définie comme une paire clé-valeur, la clé et la valeur étant chacune désignée par des guillemets simples. Chaque paire clé-valeur est définie au sein d'une structure Amazon Ion désignée par des backticks.

Le balisage des tables lors de leur création n'est actuellement pris en charge que pour les registres en mode*STANDARD* autorisations.

Valeur renvoyée

`tableId`— ID unique de la table que vous avez créée.

Baliser les tableaux au moment de la création

> **Note**
>
> Le balisage des tables lors de leur création n'est actuellement pris en charge que pour les registres en modeSTANDARD autorisations.

Vous pouvez éventuellement baliser les ressources de votre table en spécifiant des balises dans une`CREATE TABLE` instruction. Pour en savoir plus sur les identifications, consultez Balisage des ressources Amazon QLDB. L'exemple suivant crée une table nommée`Vehicle` avec la balise`environment=production`.

```
CREATE TABLE Vehicle WITH (aws_tags = `{'environment': 'production'}`)
```

Le balisage des tables lors de leur création nécessite l'accès à la fois aux`qldb:TagResource` actions`qldb:PartiQLCreateTable` et. Pour de plus amples informations sur les autorisations pour les ressources QLDB, veuillez consulterComment Amazon QLDB fonctionne avec IAM.

En attribuant des étiquettes aux ressources au moment de la création, vous pouvez supprimer la nécessité d'exécuter des scripts d'identification personnalisés après la création de ressources. Une fois qu'une table est balisée, vous pouvez contrôler l'accès à la table en fonction de ces balises. Par exemple, vous pouvez accorder un accès complet uniquement aux tables dotées d'une balise spécifique. Pour un exemple de politique JSON, consultezAccès complet à toutes les actions en fonction des balises du tableau.

Exemples

```
CREATE TABLE VehicleRegistration
```

```
CREATE TABLE Vehicle WITH (aws_tags = `{'environment': 'development'}`)
```

```
CREATE TABLE Vehicle WITH (aws_tags = `{'key1': 'value1', 'key2': 'value2'}`)
```

Exécution par programmation à l'aide du pilote

Pour savoir comment exécuter cette instruction par programmation à l'aide du pilote QLDB, consultez les didacticiels suivants dans la section Prise en main du pilote :

- Java :Dacticiel de démarrage rapide |Référence du livre de recettes
- .NET :Ddémarrage rapide |Référence de livre de cuisine
- Allez :Didacticiel de démarrage rapide |Référence de livre de recettes
- Node.js :Didacticiel de démarrage rapide |Référence de livre de cuisine
- Python :Didacticiel de démarrage rapide |Référence de livre de recettes

Commande DELETE dans Amazon QLDB

Dans Amazon QLDB, utilisez la`DELETE` commande pour marquer un document actif comme supprimé dans un tableau en créant une nouvelle révision finale du document. Cette révision finale indique que le document est supprimé. Cette opération met fin au cycle de vie d'un document, ce qui signifie qu'aucune autre révision de document avec le même identifiant de document ne peut être créée.

Cette opération est irréversible. Vous pouvez toujours consulter l'historique des révisions d'un document supprimé à l'aide duFonction historique.

> **Note**
>
> Pour savoir comment contrôler l'accès afin d'exécuter cette commande PartiQL sur des tables spécifiques, consultezCommencer à utiliser le mode d'autorisation standard dans Amazon QLDB.

Rubriques

Syntaxe

```
DELETE FROM table_name [ AS table_alias ] [ BY id_alias ]
[ WHERE condition ]
```

Paramètres

`table_name`

Nom de la table utilisateur contenant les données à supprimer. Les instructions DML ne sont prises en charge que dans la vue utilisateur par défaut. Chaque instruction ne peut être exécutée que sur une seule table.

En tant que ***table_alias***

(Facultatif) Alias défini par l'utilisateur qui couvre une table à supprimer. LeAS mot-clé est facultatif.

PAR ***id_alias***

(Facultatif) Alias défini par l'utilisateur qui se lie au champ `deid` métadonnées de chaque document du jeu de résultats. L'alias doit être déclaré dans la`FROM` clause à l'aide du`BY` mot clé. Cela est utile lorsque vous souhaitez filtrer en fonction de l'ID du document tout en interrogeant la vue utilisateur par défaut. Pour plus d'informations, veuillez consulter Utilisation de la clause BY pour interroger l'ID du document.

WHERE ***condition***

Critères de sélection des documents à supprimer.

> **Note**
>
> Si vous omettez la`WHERE` clause, tous les documents de la table sont supprimés.

Valeur renvoyée

`documentId`— L'identifiant unique de chaque document que vous avez supprimé.

Exemples

```
DELETE FROM VehicleRegistration AS r
WHERE r.VIN = '1HVBBAANXWH544237'
```

Exécution par programmation à l'aide du pilote

Pour savoir comment exécuter cette instruction par programmation à l'aide du pilote QLDB, consultez les didacticiels suivants dans la section Prise en main du pilote :

- Java :Dacticiel de démarrage rapide |Référence du livre de recettes
- .NET :Ddémarrage rapide |Référence de livre de cuisine
- Allez :Didacticiel de démarrage rapide |Référence de livre de recettes

- Node.js :Didacticiel de démarrage rapide |Référence de livre de cuisine
- Python :Didacticiel de démarrage rapide |Référence de livre de recettes

Commande DROP INDEX dans Amazon QLDB

Dans Amazon QLDB, utilisez la`DROP INDEX` commande pour supprimer un index sur une table.

> **Note**
>
> Pour savoir comment contrôler l'accès afin d'exécuter cette commande PartiQL sur des tables spécifiques, consultezCommencer à utiliser le mode d'autorisation standard dans Amazon QLDB.

Rubriques

Syntaxe

```
DROP INDEX "indexId" ON table_name WITH (purge = true)
```

> **Note**
>
> La clause`WITH (purge = true)` est obligatoire pour toutes les`DROP INDEX` instructions et`true` est actuellement la seule valeur prise en charge.
> Le mot clé`purge` est sensible à la casse et doit être entièrement en minuscules.

Paramètres

« *ID d'index* »

L'identifiant unique de l'index à supprimer, indiqué par des guillemets doubles.

SUR *table_name*

Nom de la table dont vous voulez supprimer l'index.

Valeur renvoyée

`tableId`— L'identifiant unique de la table dont vous avez supprimé l'index.

Exemples

```
DROP INDEX "4tPW3fUhaVhDinRgKRLhGU" ON VehicleRegistration WITH (purge = true)
```

Commande DROP TABLE dans Amazon QLDB

Dans Amazon QLDB, utilisez la`DROP TABLE` commande pour désactiver une table existante. Vous pouvez utiliser laTABLEAU DE DÉBALLAGE déclaration pour le réactiver. La désactivation ou la réactivation d'un tableau n'a aucun effet sur ses documents ou ses index.

> **Note**
>
> Pour savoir comment contrôler l'accès afin d'exécuter cette commande PartiQL sur des tables spécifiques, consultezCommencer à utiliser le mode d'autorisation standard dans Amazon QLDB.

Rubriques

- Syntaxe
- Paramètres
- Valeur renvoyée
- Exemples

Syntaxe

```
DROP TABLE table_name
```

Paramètres

`table_name`

Nom de la table à désactiver. La table doit déjà exister et avoir un statut`ACTIVE`.

Valeur renvoyée

`tableId`— ID unique de la table que vous avez désactivée.

Exemples

```
DROP TABLE VehicleRegistration
```

Commande FROM (INSERT, REMOVE ou SET) dans Amazon QLDB

Dans Amazon QLDB, une instruction qui commence par`FROM` est une extension PartiQL qui vous permet d'insérer et de supprimer des éléments spécifiques dans un document. Vous pouvez également utiliser cette instruction pour mettre à jour des éléments existants dans un document, de la même manière que laMISE A JOUR commande.

> **Note**
>
> Pour savoir comment contrôler l'accès afin d'exécuter cette commande PartiQL sur des tables spécifiques, consultezCommencer à utiliser le mode d'autorisation standard dans Amazon QLDB.

Rubriques

Syntaxe

À PARTIR DE L'INSERTION

Insérez un nouvel élément dans un document existant. Pour insérer un nouveau document de niveau supérieur dans un tableau, vous devez utiliserINSERT.

```
FROM table_name [ AS table_alias ] [ BY id_alias ]
[ WHERE condition ]
INSERT INTO element VALUE data [ AT key_name ]
```

À PARTIR DE SUPPRIMER

Supprimez un élément existant dans un document ou supprimez l'intégralité d'un document de niveau supérieur. Cette dernière est sémantiquement identique à laDELETE syntaxe traditionnelle.

```
FROM table_name [ AS table_alias ] [ BY id_alias ]
[ WHERE condition ]
REMOVE element
```

À PARTIR DU SET

Mettez à jour un ou plusieurs éléments d'un document. Si un élément n'existe pas, il est inséré. Du point de vue sémantique, c'est la même syntaxe que laMISE A JOUR syntaxe traditionnelle.

```
FROM table_name [ AS table_alias ] [ BY id_alias ]
[ WHERE condition ]
SET element = data [, element = data, ... ]
```

Paramètres

table_name

Nom de la table utilisateur contenant les données à modifier. Les instructions DML ne sont prises en charge que dans la vue utilisateur par défaut. Chaque instruction ne peut être exécutée que sur une seule table.

Dans cette clause, vous pouvez également inclure une ou plusieurs collections imbriquées dans la table spécifiée. Pour en savoir plus, consultez Collections imbriquées.

En tant que ***table_alias***

(Facultatif) Alias défini par l'utilisateur qui couvre une table à modifier. Tous les alias de table utilisés dans la`WHERE` clause`SETREMOVE`,`INSERT INTO`, ou doivent être déclarés dans la`FROM` clause. Le`AS` mot-clé est facultatif.

PAR ***id_alias***

(Facultatif) Alias défini par l'utilisateur qui se lie au champ de`id` métadonnées de chaque document du jeu de résultats. L'alias doit être déclaré dans la`FROM` clause à l'aide du`BY` mot clé. Cela est utile lorsque vous souhaitez filtrer en fonction de l'ID du document tout en interrogeant la vue utilisateur par défaut. Pour plus d'informations, veuillez consulter Utilisation de la clause BY pour interroger l'ID du document.

WHERE ***condition***

Critères de sélection des documents à modifier.

> **Note**
>
> Si vous omettez la`WHERE` clause, tous les documents de la table sont modifiés.

élément

Élément de document à créer ou à modifier.

data

Une nouvelle valeur pour l'élément.

Nom_clé AT

Un nom de clé à ajouter dans les documents à modifier. Vous devez spécifier le nom correspondant`VALUE` ainsi que le nom de la clé. Cela est nécessaire pour insérer une nouvelle valeur à`AT` une position spécifique dans un document.

Collections imbriquées

Bien que vous ne puissiez exécuter une instruction DML que sur une seule table, vous pouvez spécifier des collections imbriquées dans les documents de cette table en tant que sources supplémentaires. Chaque alias que vous déclarez pour une collection imbriquée peut être utilisé dans la`WHERE` clause et la`REMOVE` clause`SETINSERT INTO`, ou.

Par exemple, lesFROM sources de l'instruction suivante incluent à la fois la`VehicleRegistration` table et la`Owners.SecondaryOwners` structure imbriquée.

```
FROM VehicleRegistration r, @r.Owners.SecondaryOwners o
WHERE r.VIN = '1N4AL11D75C109151' AND o.PersonId = 'abc123'
SET o.PersonId = 'def456'
```

Cet exemple met à jour l'élément spécifique de la`SecondaryOwners` liste qui possède un`PersonId` de`'abc123'` dans le`VehicleRegistration` document qui possède un`VIN` de`'1N4AL11D75C109151'`. Cette expression vous permet de spécifier un élément d'une liste par sa valeur plutôt que par son index.

Valeur renvoyée

`documentId`— L'identifiant unique de chaque document que vous avez mis à jour ou supprimé.

Exemples

Modifiez un élément dans un document. Si l'élément n'existe pas, il est inséré.

```
FROM Vehicle AS v
WHERE v.VIN = '1N4AL11D75C109151' AND v.Color = 'Silver'
SET v.Color = 'Shiny Gray'
```

Modifiez ou insérez un élément et un filtre dans le champ `deid` métadonnées du document attribué par le système.

```
FROM Vehicle AS v BY v_id
WHERE v_id = 'documentId'
SET v.Color = 'Shiny Gray'
```

Modifiez le`PersonId` champ du premier élément de la`Owners.SecondaryOwners` liste dans un document.

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
SET r.Owners.SecondaryOwners[0].PersonId = 'abc123'
```

Supprimez un élément existant dans un document.

```
FROM Person AS p
WHERE p.GovId = '111-22-3333'
REMOVE p.Address
```

Supprimer un document entier d'un tableau.

```
FROM Person AS p
WHERE p.GovId = '111-22-3333'
REMOVE p
```

Supprimez le premier élément de la`Owners.SecondaryOwners` liste dans un document du`VehicleRegistration` tableau.

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
REMOVE r.Owners.SecondaryOwners[0]
```

Insérez`{'Mileage':26500}` en tant que paire nom-valeur de niveau supérieur dans un document du`Vehicle` tableau.

```
FROM Vehicle AS v
WHERE v.VIN = '1N4AL11D75C109151'
INSERT INTO v VALUE 26500 AT 'Mileage'
```

Ajoutez-le`{'PersonId':'abc123'}` sous forme de paire nom-valeur dans le`Owners.SecondaryOwners` champ d'un document du`VehicleRegistration` tableau. Notez qu'`Owners.SecondaryOwners`il doit déjà exister et qu'il doit s'agir d'un type de données de liste pour que cette instruction soit valide. Dans le cas contraire, le mot clé`AT` est obligatoire dans la`INSERT INTO` clause.

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
INSERT INTO r.Owners.SecondaryOwners VALUE { 'PersonId' : 'abc123' }
```

Insérer en`{'PersonId':'abc123'}` tant que premier élément de la`Owners.SecondaryOwners` liste existante d'un document.

```
FROM VehicleRegistration AS r
```

```
WHERE r.VIN = '1N4AL11D75C109151'
INSERT INTO r.Owners.SecondaryOwners VALUE {'PersonId' : 'abc123'} AT 0
```

Ajoutez plusieurs paires nom-valeur à la`Owners.SecondaryOwners` liste existante d'un document.

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
INSERT INTO r.Owners.SecondaryOwners << {'PersonId' : 'abc123'}, {'PersonId' :
 'def456'} >>
```

Exécution par programmation à l'aide du pilote

Pour savoir comment exécuter cette instruction par programmation à l'aide du pilote QLDB, consultez les didacticiels suivants dans la section Prise en main du pilote :

- Java :Dacticiel de démarrage rapide |Référence du livre de recettes
- .NET :Ddémarrage rapide |Référence de livre de cuisine
- Allez :Didacticiel de démarrage rapide |Référence de livre de recettes
- Node.js :Didacticiel de démarrage rapide |Référence de livre de cuisine
- Python :Didacticiel de démarrage rapide |Référence de livre de recettes

Commande INSERT dans Amazon QLDB

Dans Amazon QLDB, utilisez la`INSERT` commande pour ajouter un ou plusieurs documents Amazon Ion à un tableau.

> **Note**
>
> Pour savoir comment contrôler l'accès afin d'exécuter cette commande PartiQL sur des tables spécifiques, consultezCommencer à utiliser le mode d'autorisation standard dans Amazon QLDB.

Rubriques

- Syntaxe
- Paramètres
- Valeur renvoyée

- Exemples
- Exécution par programmation à l'aide du pilote

Syntaxe

Insérez un document unique.

```
INSERT INTO table_name VALUE document
```

Insérez plusieurs documents.

```
INSERT INTO table_name << document, document, ... >>
```

Paramètres

table_name

Nom de la table utilisateur dans laquelle vous souhaitez insérer les données. La table doit déjà exister. Les instructions DML ne sont prises en charge que dans la vue utilisateur par défaut.

document

Un document QLDB valide. Vous devez spécifier au moins un document. Les documents multiples doivent être séparés par des virgules.

Le document doit être indiqué par des accolades (`{...}`).

Chaque nom de champ du document est un symbole ionique distinguant majuscules et minuscules qui peut être indiqué par des guillemets simples (`'...'`) dans PartiQL.

Les valeurs de chaîne sont également indiquées par des apostrophes (`'...'`) dans PartiQL.

Tous les littéraux ioniques peuvent être désignés par des backticks (`` `...` ``).

> **Note**
>
> Les crochets doubles (`<<...>>`) indiquent une collection non ordonnée (appelée sac dans PartiQL) et ne sont obligatoires que si vous souhaitez insérer plusieurs documents.

Valeur renvoyée

`documentId`— L'identifiant unique de chaque document que vous avez inséré.

Exemples

Insérez un document unique.

```
INSERT INTO VehicleRegistration VALUE
{
    'VIN' : 'KM8SRDHF6EU074761', --string
    'RegNum' : 1722, --integer
    'State' : 'WA',
    'City' : 'Kent',
    'PendingPenaltyTicketAmount' : 130.75, --decimal
    'Owners' : { --nested struct
        'PrimaryOwner' : { 'PersonId': '294jJ3YUoH1IEEm8GSabOs' },
        'SecondaryOwners' : [ --list of structs
            { 'PersonId' : '1nmeDdLo3AhGswBtyM1eYh' },
            { 'PersonId': 'IN7MvYtUjkp1GMZu0F6CG9' }
        ]
    },
    'ValidFromDate' : `2017-09-14T`, --Ion timestamp literal with day precision
    'ValidToDate' : `2020-06-25T`
}
```

Cette instruction renvoie l'identifiant unique du document que vous avez inséré, comme suit.

```
{
    documentId: "2kKuOPNB07D2iTPBrUTWGl"
}
```

Insérez plusieurs documents.

```
INSERT INTO Person <<
{
    'FirstName' : 'Raul',
    'LastName' : 'Lewis',
    'DOB' : `1963-08-19T`,
    'GovId' : 'LEWISR261LL',
    'GovIdType' : 'Driver License',
```

```
    'Address' : '1719 University Street, Seattle, WA, 98109'
},
{
    'FirstName' : 'Brent',
    'LastName' : 'Logan',
    'DOB' : `1967-07-03T`,
    'GovId' : 'LOGANB486CG',
    'GovIdType' : 'Driver License',
    'Address' : '43 Stockert Hollow Road, Everett, WA, 98203'
},
{
    'FirstName' : 'Alexis',
    'LastName' : 'Pena',
    'DOB' : `1974-02-10T`,
    'GovId' : '744 849 301',
    'GovIdType' : 'SSN',
    'Address' : '4058 Melrose Street, Spokane Valley, WA, 99206'
}
>>
```

Cette instruction renvoie l'identifiant unique de chaque document que vous avez inséré, comme suit.

```
{
    documentId: "6WXzLscsJ3bDWW97Dy8nyp"
},
{
    documentId: "35e0ToZyTGJ7LGvcwrkX65"
},
{
    documentId: "BVHPcH612o7JROQ4yP8jiH"
}
```

Exécution par programmation à l'aide du pilote

Pour savoir comment exécuter cette instruction par programmation à l'aide du pilote QLDB, consultez les didacticiels suivants dans la section Prise en main du pilote :

- Java :Dacticiel de démarrage rapide |Référence du livre de recettes
- .NET :Ddémarrage rapide |Référence de livre de cuisine
- Allez :Didacticiel de démarrage rapide |Référence de livre de recettes
- Node.js :Didacticiel de démarrage rapide |Référence de livre de cuisine

- Python :Didacticiel de démarrage rapide |Référence de livre de recettes

Commande SELECT dans Amazon QLDB

Dans Amazon QLDB, utilisez la`SELECT` commande pour récupérer les données d'une ou de plusieurs tables. Chaque`SELECT` requête dans QLDB est traitée dans le cadre d'une transaction et est soumise à un délai d'expiration de transaction.

L'ordre des résultats n'est pas spécifique et peut varier pour chaque`SELECT` requête. Vous ne devez pas vous fier à l'ordre des résultats pour aucune requête dans QLDB.

Pour savoir comment contrôler l'accès afin d'exécuter cette commande PartiQL sur des tables spécifiques, consultezCommencer à utiliser le mode d'autorisation standard dans Amazon QLDB.

> **Warning**
>
> Lorsque vous exécutez une requête dans QLDB sans recherche indexée, elle appelle une analyse complète de la table. PartiQL prend en charge de telles requêtes car il est compatible avec SQL. Toutefois, n'exécutez pas d'analyses de tables pour des cas d'utilisation en production dans QLDB. Les analyses de tables peuvent entraîner des problèmes de performances sur des tables de grande taille, notamment des conflits de simultanéité et des délais de transaction.
>
> Pour éviter de scanner des tables, vous devez exécuter des instructions avec une clause de`WHERE` prédicat à l'aide d'un opérateur d'égalité sur un champ indexé ou un identifiant de document, par exemple,`WHERE indexedField = 123` ou`WHERE indexedField IN (456, 789)`. Pour plus d'informations, veuillez consulter Optimisation des performances des données.

Rubriques

- Syntaxe
- Paramètres
- Jointures
- Limites des requêtes imbriquées
- Exemples
- Exécution par programmation à l'aide du pilote

Syntaxe

```
SELECT [ VALUE ] expression [ AS field_alias ] [, expression, ... ]
FROM source [ AS source_alias ] [ AT idx_alias ] [ BY id_alias ] [, source, ... ]
[ WHERE condition ]
```

Paramètres

VALUE

Un qualificatif pour votre expression qui oblige la requête à renvoyer la valeur du type de données brut, plutôt que la valeur encapsulée dans une structure de tuple.

expression

Projection formée à partir du* caractère générique ou d'une liste de projection d'un ou de plusieurs champs de documents de l'ensemble de résultats. Une expression peut être constituée d'appels à Références Références Références de PartiQL ou de champs modifiés par Opérateurs PartiQL.

AS ***field_alias***

(Facultatif) Alias temporaire défini par l'utilisateur pour le champ utilisé dans le jeu de résultats final. LeAS mot-clé est facultatif.

Si vous ne spécifiez pas d'alias pour une expression qui n'est pas un simple nom de champ, le jeu de résultats applique un nom par défaut à ce champ.

À PARTIR DE ***LA SOURCE***

Une source à interroger. Les seules sources actuellement prises en charge sont les noms de tables, les jointures internes entre les tables, lesSELECT requêtes imbriquées (soumises àLimites des requêtes imbriquées) et les appels de fonctions d'historique pour une table.

Vous devez spécifier au moins une source. Les sources multiples doivent être séparées par des virgules.

En tant que ***source_alias***

(Facultatif) Alias défini par l'utilisateur qui couvre la source à interroger. Tous les alias de source utilisés dans laWHERE clauseSELECT OR doivent être déclarés dans laFROM clause. LeAS mot-clé est facultatif.

AT *idx_alias*

(Facultatif) Alias défini par l'utilisateur qui se lie au numéro d'index (ordinal) de chaque élément d'une liste provenant de la source. L'alias doit être déclaré dans la`FROM` clause à l'aide du`AT` mot clé.

PAR *id_alias*

(Facultatif) Alias défini par l'utilisateur qui se lie au champ `id` métadonnées de chaque document du jeu de résultats. L'alias doit être déclaré dans la`FROM` clause à l'aide du`BY` mot clé. Cela est utile lorsque vous souhaitez projeter ou filtrer l'ID du document tout en interrogeant la vue utilisateur par défaut. Pour plus d'informations, veuillez consulter Utilisation de la clause BY pour interroger l'ID du document.

WHERE *condition*

Les critères de sélection et les critères de jointure (le cas échéant) pour la requête.

> **Note**
>
> Si vous omettez la`WHERE` clause, tous les documents de la table sont extraits.

Jointures

Seules les jointures internes sont actuellement prises en charge. Vous pouvez écrire des requêtes de jointure internes à l'aide de la`INNER JOIN` clause explicite, comme suit. Dans cette syntaxe,`JOIN` doit être associé à`ON`, et le`INNER` mot clé est facultatif.

```
SELECT expression
FROM table1 AS t1 [ INNER ] JOIN table2 AS t2
ON t1.element = t2.element
```

Vous pouvez également écrire des jointures internes en utilisant la syntaxe implicite, comme suit.

```
SELECT expression
FROM table1 AS t1, table2 AS t2
WHERE t1.element = t2.element
```

Limites des requêtes imbriquées

Vous pouvez écrire des requêtes imbriquées (sous-requêtes) dans des`SELECT` expressions et dans`FROM` des sources. La principale restriction est que seule la requête la plus externe peut accéder à l'environnement de base de données global. Par exemple, supposons que vous ayez un registre avec des tables`VehicleRegistration` et`Person`. La requête imbriquée suivante n'est pas valide car l'interne`SELECT` essaie d'y accéder`Person`.

```
SELECT r.VIN,
    (SELECT p.PersonId FROM Person AS p WHERE p.PersonId =
 r.Owners.PrimaryOwner.PersonId) AS PrimaryOwner
FROM VehicleRegistration AS r
```

Alors que la requête imbriquée suivante est valide.

```
SELECT r.VIN, (SELECT o.PrimaryOwner.PersonId FROM @r.Owners AS o) AS PrimaryOwner
FROM VehicleRegistration AS r
```

Exemples

La requête suivante affiche un caractère`SELECT` générique de base avec une clause de`WHERE` prédicat standard qui utilise l'`IN`opérateur.

```
SELECT * FROM Vehicle
WHERE VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

Ce qui suit montre`SELECT` les projections avec un filtre de chaîne.

```
SELECT FirstName, LastName, Address
FROM Person
WHERE Address LIKE '%Seattle%'
AND GovId = 'LEWISR261LL'
```

Ce qui suit montre une sous-requête corrélée qui aplatit les données imbriquées. Notez que le`@` caractère est techniquement facultatif ici. Mais cela indique explicitement que vous souhaitez que la`Owners` structure y soit imbriquée`VehicleRegistration`, et non qu'une collection différente soit nommée`Owners` (s'il en existe une). Pour plus de contexte, consultezDonnées imbriquées le chapitre Utilisation des données et de l'historique.

```
SELECT
```

```
    r.VIN,
    o.SecondaryOwners
FROM
    VehicleRegistration AS r, @r.Owners AS o
WHERE
    r.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

Ce qui suit montre une sous-requête de la`SELECT` liste qui projette des données imbriquées et une jointure interne implicite.

```
SELECT
    v.Make,
    v.Model,
    (SELECT VALUE o.PrimaryOwner.PersonId FROM @r.Owners AS o) AS PrimaryOwner
FROM
    VehicleRegistration AS r, Vehicle AS v
WHERE
    r.VIN = v.VIN AND r.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

Ce qui suit montre une jointure interne explicite.

```
SELECT
    v.Make,
    v.Model,
    r.Owners
FROM
    VehicleRegistration AS r JOIN Vehicle AS v
ON
    r.VIN = v.VIN
WHERE
    r.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

Ce qui suit montre une projection du champ de`id` métadonnées du document, à l'aide de la`BY` clause.

```
SELECT
    r_id,
    r.VIN
FROM
    VehicleRegistration AS r BY r_id
WHERE
```

```
    r_id = 'documentId'
```

Ce qui suit utilise laBY clause pour joindre les`Person` tables`DriversLicense` et respectivement dans leurs champs`PersonId` et`id` dans les champs du document.

```
SELECT * FROM DriversLicense AS d INNER JOIN Person AS p BY pid
ON d.PersonId = pid
WHERE pid = 'documentId'
```

Ce qui suit utilise lePoint de vue engagé pour joindre les`Person` tables`DriversLicense` et respectivement`id` dans leurs champs`PersonId` et ceux du document.

```
SELECT * FROM DriversLicense AS d INNER JOIN _ql_committed_Person AS cp
ON d.PersonId = cp.metadata.id
WHERE cp.metadata.id = 'documentId'
```

Ce qui suit renvoie le numéro d'index (ordinal)`PersonId` et le numéro d'index (ordinal) de chaque personne de la`Owners.SecondaryOwners` liste pour un document dans le tableau`VehicleRegistration`.

```
SELECT s.PersonId, owner_idx
FROM VehicleRegistration AS r, @r.Owners.SecondaryOwners AS s AT owner_idx
WHERE r.VIN = 'KM8SRDHF6EU074761'
```

Exécution par programmation à l'aide du pilote

Pour savoir comment exécuter cette instruction par programmation à l'aide du pilote QLDB, consultez les didacticiels suivants dans la section Prise en main du pilote :

- Java :Dacticiel de démarrage rapide |Référence du livre de recettes
- .NET :Ddémarrage rapide |Référence de livre de cuisine
- Allez :Didacticiel de démarrage rapide |Référence de livre de recettes
- Node.js :Didacticiel de démarrage rapide |Référence de livre de cuisine
- Python :Didacticiel de démarrage rapide |Référence de livre de recettes

Commande UPDATE dans Amazon QLDB

Dans Amazon QLDB, utilisez laUPDATE commande pour modifier la valeur d'un ou de plusieurs éléments dans un document. Si un élément n'existe pas, il est inséré.

Vous pouvez également utiliser cette commande pour insérer et supprimer de manière explicite des éléments spécifiques dans un document, à l'instarDE (INSÉRER, SUPPRIMER ou DÉFINIR) des instructions.

> **Note**
>
> Pour savoir comment contrôler l'accès afin d'exécuter cette commande PartiQL sur des tables spécifiques, consultezCommencer à utiliser le mode d'autorisation standard dans Amazon QLDB.

Rubriques

- Syntaxe
- Paramètres
- Valeur renvoyée
- Exemples
- Exécution par programmation à l'aide du pilote

Syntaxe

ENSEMBLE DE MISES À JOUR

Mettez à jour un ou plusieurs éléments d'un document. Si un élément n'existe pas, il est inséré. Du point de vue sémantique, c'est la même chose que l'instruction FROM-SET.

```
UPDATE table_name [ AS table_alias ] [ BY id_alias ]
SET element = data [, element = data, ... ]
[ WHERE condition ]
```

METTRE À JOUR-INSÉRER

Insérez un nouvel élément dans un document existant. Pour insérer un nouveau document de niveau supérieur dans un tableau, vous devez utiliserINSERT.

```
UPDATE table_name [ AS table_alias ] [ BY id_alias ]
INSERT INTO element VALUE data [ AT key_name ]
[ WHERE condition ]
```

METTRE À JOUR-SUPPRIMER

Supprimez un élément existant dans un document ou supprimez l'intégralité d'un document de niveau supérieur. Cette dernière est sémantiquement identique à laDELETE syntaxe traditionnelle.

```
UPDATE table_name [ AS table_alias ] [ BY id_alias ]
REMOVE element
[ WHERE condition ]
```

Paramètres

table_name

Nom de la table utilisateur contenant les données à modifier. Les instructions DML ne sont prises en charge que dans la vue utilisateur par défaut. Chaque instruction ne peut être exécutée que sur une seule table.

En tant que ***table_alias***

(Facultatif) Alias défini par l'utilisateur qui couvre une table à mettre à jour. LeAS mot-clé est facultatif.

PAR ***id_alias***

(Facultatif) Alias défini par l'utilisateur qui se lie au champ de`id` métadonnées de chaque document du jeu de résultats. L'alias doit être déclaré dans la`UPDATE` clause à l'aide duBY mot clé. Cela est utile lorsque vous souhaitez filtrer en fonction de l'ID du document tout en interrogeant la vue utilisateur par défaut. Pour plus d'informations, veuillez consulter Utilisation de la clause BY pour interroger l'ID du document.

élément

Élément de document à créer ou à modifier.

data

Une nouvelle valeur pour l'élément.

Nom_clé AT

Un nom de clé à ajouter dans les documents à modifier. Vous devez spécifier le nom correspondantVALUE ainsi que le nom de la clé. Cela est nécessaire pour insérer une nouvelle valeur àAT une position spécifique dans un document.

WHERE ***condition***

Critères de sélection des documents à modifier.

> **Note**
>
> Si vous omettez laWHERE clause, tous les documents de la table sont modifiés.

Valeur renvoyée

`documentId`— L'identifiant unique de chaque document que vous avez mis à jour.

Exemples

Mettez à jour un champ dans un document. Si le champ n'existe pas, il est inséré.

```
UPDATE Person AS p
SET p.LicenseNumber = 'HOLLOR123ZZ'
WHERE p.GovId = '111-22-3333'
```

Filtrez sur le champ `deid` métadonnées du document attribué par le système.

```
UPDATE Person AS p BY pid
SET p.LicenseNumber = 'HOLLOR123ZZ'
WHERE pid = 'documentId'
```

Remplacez un document entier.

```
UPDATE Person AS p
SET p = {
    'FirstName' : 'Rosemarie',
    'LastName' : 'Holloway',
    'DOB' : `1977-06-18T`,
    'GovId' : '111-22-3333',
```

```
    'GovIdType' : 'Driver License',
    'Address' : '4637 Melrose Street, Ellensburg, WA, 98926'
}
WHERE p.GovId = '111-22-3333'
```

Modifiez le`PersonId` champ du premier élément de la`Owners.SecondaryOwners` liste dans un document.

```
UPDATE VehicleRegistration AS r
SET r.Owners.SecondaryOwners[0].PersonId = 'abc123'
WHERE r.VIN = '1N4AL11D75C109151'
```

Insérez`{'Mileage':26500}` en tant que paire nom-valeur de niveau supérieur dans un document du`Vehicle` tableau.

```
UPDATE Vehicle AS v
INSERT INTO v VALUE 26500 AT 'Mileage'
WHERE v.VIN = '1N4AL11D75C109151'
```

Ajoutez-le`{'PersonId':'abc123'}` sous forme de paire nom-valeur dans le`Owners.SecondaryOwners` champ d'un document du`VehicleRegistration` tableau. Notez qu'`Owners.SecondaryOwners`il doit déjà exister et qu'il doit s'agir d'un type de données de liste pour que cette instruction soit valide. Dans le cas contraire, le mot clé`AT` est obligatoire dans la`INSERT INTO` clause.

```
UPDATE VehicleRegistration AS r
INSERT INTO r.Owners.SecondaryOwners VALUE { 'PersonId' : 'abc123' }
WHERE r.VIN = '1N4AL11D75C109151'
```

Insérer en`{'PersonId':'abc123'}` tant que premier élément de la`Owners.SecondaryOwners` liste existante d'un document.

```
UPDATE VehicleRegistration AS r
INSERT INTO r.Owners.SecondaryOwners VALUE {'PersonId' : 'abc123'} AT 0
WHERE r.VIN = '1N4AL11D75C109151'
```

Ajoutez plusieurs paires nom-valeur à la`Owners.SecondaryOwners` liste existante d'un document.

```
UPDATE VehicleRegistration AS r
```

```
INSERT INTO r.Owners.SecondaryOwners << {'PersonId' : 'abc123'}, {'PersonId' :
 'def456'} >>
WHERE r.VIN = '1N4AL11D75C109151'
```

Supprimez un élément existant dans un document.

```
UPDATE Person AS p
REMOVE p.Address
WHERE p.GovId = '111-22-3333'
```

Supprimer un document entier d'un tableau.

```
UPDATE Person AS p
REMOVE p
WHERE p.GovId = '111-22-3333'
```

Supprimez le premier élément de la`Owners.SecondaryOwners` liste dans un document du`VehicleRegistration` tableau.

```
UPDATE VehicleRegistration AS r
REMOVE r.Owners.SecondaryOwners[0]
WHERE r.VIN = '1N4AL11D75C109151'
```

Exécution par programmation à l'aide du pilote

Pour savoir comment exécuter cette instruction par programmation à l'aide du pilote QLDB, consultez les didacticiels suivants dans la section Prise en main du pilote :

- Java :Dacticiel de démarrage rapide |Référence du livre de recettes
- .NET :Ddémarrage rapide |Référence de livre de cuisine
- Allez :Didacticiel de démarrage rapide |Référence de livre de recettes
- Node.js :Didacticiel de démarrage rapide |Référence de livre de cuisine
- Python :Didacticiel de démarrage rapide |Référence de livre de recettes

Commande UNDROP TABLE dans Amazon QLDB

Dans Amazon QLDB, utilisez la`UNDROP TABLE` commande pour réactiver une table que vous avez précédemment supprimée (c'est-à-dire désactivée). La désactivation ou la réactivation d'un tableau n'a aucun effet sur ses documents ou ses index.

> **Note**
>
> Pour savoir comment contrôler l'accès afin d'exécuter cette commande PartiQL sur des tables spécifiques, consultezCommencer à utiliser le mode d'autorisation standard dans Amazon QLDB.

Rubriques

Syntaxe

```
UNDROP TABLE "tableId"
```

Paramètres

« ***ID de table*** »

L'identifiant unique de la table à réactiver, indiqué par des guillemets doubles.

La table doit avoir déjà été supprimée, ce qui signifie qu'elle existe dans la table du catalogue système`information_schema.user_tables` et qu'elle a le statut`INACTIVE`. Il ne doit pas non plus exister de table active portant le même nom.

Valeur renvoyée

`tableId`— ID unique de la table que vous avez réactivée.

Exemples

```
UNDROP TABLE "5PLf9SXwndd63lPaSIa0O6"
```

Fonctions partiQL dans Amazon QLDB

PartiQL dans Amazon QLDB en charge les variantes intégrées suivantes des fonctions standard SQL.

> **Note**
>
> Les fonctions SQL qui ne figurent pas dans cette liste ne sont actuellement pas prises en charge dans QLDB.
> Cette référence de fonction est basée sur la documentation PartiQL Built-Functions.

Type de propagation inconnu (nul et manquant)

Sauf indication contraire, toutes ces fonctions propagent des valeurs d'argument nulles et manquantes. La propagation de`NULL` ou`MISSING` est définie comme renvoyant`NULL` si un argument de fonction est soit soit`NULL` soit`MISSING`. Voici des exemples de cette propagation.

```
CHAR_LENGTH(null)    -- null
CHAR_LENGTH(missing) -- null (also returns null)
```

Fonctions d'agrégation

- AVG
- COUNT
- MAX
- MIN
- SIZE
- SUM

Fonctions conditionnelles

- COALESCE
- EXISTS
- NULLIF

Fonctions de date et d'heure

- DATE_ADD
- DATE_DIFF
- EXTRACT
- UTCNOW

Fonctions scalaires

- TXID

Fonctions de chaîne

- CHAR_LENGTH
- CHARACTER_LENGTH
- LOWER
- SUBSTRING
- TRIM
- UPPER

Fonctions de formatage des types de données

- CAST
- TO_STRING
- TO_TIMESTAMP

Fonction AVG dans Amazon QLDB

Dans Amazon QLDB, utilisez laAVG fonction pour renvoyer la moyenne (moyenne arithmétique) des valeurs d'expression d'entrée. Cette fonction fonctionne avec des valeurs numériques et ignore les valeurs nulles ou manquantes.

Syntaxe

```
AVG ( expression )
```

Arguments

expression

Nom ou expression d'un type de données numérique sur lequel la fonction opère.

Types de données

Types d'arguments pris en charge :

- `int`
- `decimal`
- `float`

Type de retour :`decimal`

Exemples

```
SELECT AVG(r.PendingPenaltyTicketAmount) FROM VehicleRegistration r  -- 147.19
SELECT AVG(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>         -- 2.
```

Références Références

- COUNT
- MAX
- MIN
- SIZE

- SUM

Fonction CAST dans Amazon QLDB

Dans Amazon QLDB, utilisez laCAST fonction pour évaluer une expression donnée en valeur et convertir cette valeur en un type de données cible spécifié. Si la conversion ne peut pas être faite, la fonction renvoie.

Syntaxe

```
CAST ( expression AS type )
```

Arguments

expression

Nom ou expression ayant pour valeur une valeur que la fonction convertit. La conversion de valeurs null renvoie des valeurs null. Ce paramètre peut être n'importe lequel des paramètres pris en chargeTypes de données.

type

Nom du type de données cible pour la conversion. Ce paramètre peut être l'un des paramètres pris en chargeTypes de données.

Type de retour

Type de données spécifié par l'argument *type*.

Exemples

Les exemples suivants montrent la propagation de types inconnus (NULLouMISSING).

```
CAST(null    AS null)    -- null
CAST(missing AS null)    -- null
CAST(missing AS missing) -- missing
CAST(null    AS missing) -- missing
CAST(null    AS boolean) -- null (null AS any data type name results in null)
CAST(missing AS boolean) -- missing (missing AS any data type name results in missing)
```

Toute valeur qui n'est pas un type inconnu ne peut pas être convertie enNULL ouMISSING.

```
CAST(true AS null)    -- error
CAST(true AS missing) -- error
CAST(1    AS null)    -- error
CAST(1    AS missing) -- error
```

Les exemples suivants illustrent le casting`AS boolean`.

```
CAST(true      AS boolean) -- true no-op
CAST(0         AS boolean) -- false
CAST(1         AS boolean) -- true
CAST(`1e0`     AS boolean) -- true (float)
CAST(`1d0`     AS boolean) -- true (decimal)
CAST('a'       AS boolean) -- false
CAST('true'    AS boolean) -- true (SqlName string 'true')
CAST(``'true'`` AS boolean) -- true (Ion symbol ``'true'``)
CAST(``'false'`` AS boolean) -- false (Ion symbol ``'false'``)
```

Les exemples suivants illustrent le casting`AS integer`.

```
CAST(true   AS integer) -- 1
CAST(false  AS integer) -- 0
CAST(1      AS integer) -- 1
CAST(`1d0`  AS integer) -- 1
CAST(`1d3`  AS integer) -- 1000
CAST(1.00   AS integer) -- 1
CAST(1.45   AS integer) -- 1
CAST(1.75   AS integer) -- 1
CAST('12'   AS integer) -- 12
CAST('aa'   AS integer) -- error
CAST(``'22'`` AS integer) -- 22
CAST(``'x'``  AS integer) -- error
```

Les exemples suivants illustrent le casting`AS float`.

```
CAST(true   AS float) -- 1e0
CAST(false  AS float) -- 0e0
CAST(1      AS float) -- 1e0
CAST(`1d0`  AS float) -- 1e0
CAST(`1d3`  AS float) -- 1000e0
CAST(1.00   AS float) -- 1e0
CAST('12'   AS float) -- 12e0
CAST('aa'   AS float) -- error
```

```
CAST(``'22'`` AS float) -- 22e0
CAST(``'x'``  AS float) -- error
```

Les exemples suivants illustrent le castingAS `decimal`.

```
CAST(true   AS decimal) -- 1.
CAST(false  AS decimal) -- 0.
CAST(1      AS decimal) -- 1.
CAST(`1d0`  AS decimal) -- 1. (REPL printer serialized to 1.)
CAST(`1d3`  AS decimal) -- 1d3
CAST(1.00   AS decimal) -- 1.00
CAST('12'   AS decimal) -- 12.
CAST('aa'   AS decimal) -- error
CAST(``'22'`` AS decimal) -- 22.
CAST(``'x'``  AS decimal) -- error
```

Les exemples suivants illustrent le castingAS `timestamp`.

```
CAST(`2001T`                      AS timestamp) -- 2001T
CAST('2001-01-01T'                AS timestamp) -- 2001-01-01T
CAST(``'2010-01-01T00:00:00.000Z'`` AS timestamp) -- 2010-01-01T00:00:00.000Z
CAST(true                         AS timestamp) -- error
CAST(2001                         AS timestamp) -- error
```

Les exemples suivants illustrent le castingAS `symbol`.

```
CAST(``'xx'``                   AS symbol) -- xx (``'xx'`` is an Ion symbol)
CAST('xx'                       AS symbol) -- xx ('xx' is a string)
CAST(42                         AS symbol) -- '42'
CAST(`1e0`                      AS symbol) -- '1.0'
CAST(`1d0`                      AS symbol) -- '1'
CAST(true                       AS symbol) -- 'true'
CAST(false                      AS symbol) -- 'false'
CAST(`2001T`                    AS symbol) -- '2001T'
CAST(`2001-01-01T00:00:00.000Z` AS symbol) -- '2001-01-01T00:00:00.000Z'
```

Les exemples suivants illustrent le castingAS `string`.

```
CAST(``'xx'``                   AS string) -- "xx" (``'xx'`` is an Ion symbol)
CAST('xx'                       AS string) -- "xx" ('xx' is a string)
CAST(42                         AS string) -- "42"
```

```
CAST(`1e0`                      AS string) -- "1.0"
CAST(`1d0`                      AS string) -- "1"
CAST(true                       AS string) -- "true"
CAST(false                      AS string) -- "false"
CAST(`2001T`                    AS string) -- "2001T"
CAST(`2001-01-01T00:00:00.000Z` AS string) -- "2001-01-01T00:00:00.000Z"
```

Les exemples suivants illustrent le castingAS `struct`.

```
CAST(`{ a: 1 }` AS struct) -- {a:1}
CAST(true       AS struct) -- err
```

Les exemples suivants illustrent le castingAS `list`.

```
CAST(`[1, 2, 3]`        AS list) -- [1,2,3]
CAST(<<'a', { 'b':2 }>> AS list) -- ["a",{'b':2}]
CAST({ 'b':2 }          AS list) -- error
```

Les exemples suivants sont des instructions exécutables qui incluent certains des exemples précédents.

```
SELECT CAST(true AS integer) FROM << 0 >>            -- 1
SELECT CAST('2001-01-01T' AS timestamp) FROM << 0 >> -- 2001-01-01T
SELECT CAST('xx' AS symbol) FROM << 0 >>             -- xx
SELECT CAST(42 AS string) FROM << 0 >>               -- "42"
```

Références Références

- TO_STRING
- TO_TIMESTAMP

Fonction CHAR_LENGTH dans Amazon QLDB

Dans Amazon QLDB, utilisez laCHAR_LENGTH fonction pour renvoyer le nombre de caractères de la chaîne spécifiée, le caractère étant défini comme un point de code Unicode unique.

Syntaxe

```
CHAR_LENGTH ( string )
```

`CHAR_LENGTH`est un synonyme de[Fonction CHARACTER_LENGTH dans Amazon QLDB](#).

Arguments

`string`

Nom du champ ou expression du type`string` de données évalué par la fonction.

Type de retour

`int`

Exemples

```
SELECT CHAR_LENGTH('') FROM << 0 >>        -- 0
SELECT CHAR_LENGTH('abcdefg') FROM << 0 >> -- 7
SELECT CHAR_LENGTH('e#') FROM << 0 >>      -- 2 (because 'e#' is two code points: the
 letter 'e' and combining character U+032B)
```

Références Références

- LOWER
- SUBSTRING
- TRIM
- UPPER

Fonction CHARACTER_LENGTH dans Amazon QLDB

Synonyme de la fonction `CHAR_LENGTH`.

Consultez [Fonction CHAR_LENGTH dans Amazon QLDB](#).

Fonction COALESCE dans Amazon QLDB

Dans Amazon QLDB, à partir d'une liste d'un ou de plusieurs arguments, utilisez la`COALESCE` fonction pour évaluer les arguments dans l'ordre de gauche à droite et renvoyer la première valeur qui n'est pas de type inconnu (`NULL`ou`MISSING`). Si tous les types d'arguments sont inconnus, le résultat est`NULL`.

La`COALESCE` fonction ne se propage pas`NULL` et`MISSING`.

Syntaxe

```
COALESCE ( expression [, expression, ... ] )
```

Arguments

`expression`

La liste d'un ou de plusieurs noms de champs ou expressions que la fonction évalue. Chaque argument peut être n'importe lequel des arguments pris en chargeTypes de données.

Type de retour

Tout type de données pris en charge. Le type de retour est`NULL` soit identique, soit identique au type de la première expression qui donne une valeur non nulle et non manquante.

Exemples

```
SELECT COALESCE(1, null) FROM << 0 >>          -- 1
SELECT COALESCE(null, null, 1) FROM << 0 >>    -- 1
SELECT COALESCE(null, 'string') FROM << 0 >>  -- "string"
```

Références Références

- EXISTS
- NULLIF

Fonction COUNT dans Amazon QLDB

Dans Amazon QLDB, utilisez la`COUNT` fonction pour renvoyer le nombre de documents définis par l'expression donnée. Cette fonction se décline en deux variantes :

- `COUNT(*)`— Compte tous les documents de la table cible, qu'ils contiennent ou non des valeurs nulles ou manquantes.
- `COUNT(expression)`— Calcule le nombre de documents contenant des valeurs non nulles dans un champ ou une expression spécifique et existant.

> **Warning**
>
> La`COUNT` fonction n'est pas optimisée, nous ne recommandons donc pas de l'utiliser sans recherche indexée. Lorsque vous exécutez une requête dans QLDB sans recherche indexée, elle appelle une analyse complète de la table. Cela peut entraîner des problèmes de performances sur des tables de grande taille, notamment des conflits de simultanéité et des délais de transaction.
>
> Pour éviter de scanner des tables, vous devez exécuter des instructions avec une clause de`WHERE` prédicat à l'aide d'un opérateur d'égalité (`=`ou`IN`) sur un champ indexé ou un identifiant de document. Pour plus d'informations, veuillez consulter Optimisation des performances des données.

Syntaxe

```
COUNT ( * | expression )
```

Arguments

`expression`

Nom ou expression sur laquelle la fonction opère. Ce paramètre peut être n'importe lequel des paramètres pris en chargeTypes de données.

Type de retour

`int`

Exemples

```
SELECT COUNT(*) FROM VehicleRegistration r WHERE r.LicensePlateNumber = 'CA762X' -- 1
SELECT COUNT(r.VIN) FROM Vehicle r WHERE r.VIN = '1N4AL11D75C109151'              -- 1
SELECT COUNT(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>                   -- 3
```

Références Références

- AVG
- MAX

- MIN
- SIZE
- SUM

Fonction DATE_ADD dans Amazon QLDB

Dans Amazon QLDB, utilisez laDATE_ADD fonction pour incrémenter une valeur d'horodatage donnée selon un intervalle spécifié.

Syntaxe

```
DATE_ADD( datetimepart, interval, timestamp )
```

Arguments

date/heure (partie)

Références sur lesquelles la fonction opère. Ce paramètre peut avoir l'une des valeurs suivantes :

- `year`
- `month`
- `day`
- `hour`
- `minute`
- `second`

interval

Entier qui spécifie l'intervalle à ajouter à l'*horodatage* donné. Un nombre entier négatif soustrait l'intervalle.

timestamp

Nom du champ ou expression du type de donnéestimestamp que la fonction incrémente.

Une valeur littérale d'horodatage Ion peut être indiquée par des crochets inverses (`` `...` ``). Pour plus de détails sur le formatage et des exemples de valeurs d'horodatage, consultez la section Horodatages du document de spécifications Amazon Ion.

Type de retour

`timestamp`

Exemples

```
DATE_ADD(year, 5, `2010-01-01T`)                -- 2015-01-01T
DATE_ADD(month, 1, `2010T`)                     -- 2010-02T (result adds precision as
 necessary)
DATE_ADD(month, 13, `2010T`)                    -- 2011-02T (2010T is equivalent to
 2010-01-01T00:00:00.000Z)
DATE_ADD(day, -1, `2017-01-10T`)                -- 2017-01-09T
DATE_ADD(hour, 1, `2017T`)                      -- 2017-01-01T01:00Z
DATE_ADD(hour, 1, `2017-01-02T03:04Z`)          -- 2017-01-02T04:04Z
DATE_ADD(minute, 1, `2017-01-02T03:04:05.006Z`) -- 2017-01-02T03:05:05.006Z
DATE_ADD(second, 1, `2017-01-02T03:04:05.006Z`) -- 2017-01-02T03:04:06.006Z

-- Runnable statements
SELECT DATE_ADD(year, 5, `2010-01-01T`) FROM << 0 >> -- 2015-01-01T
SELECT DATE_ADD(day, -1, `2017-01-10T`) FROM << 0 >> -- 2017-01-09T
```

Références Références

- DATE_DIFF
- EXTRACT
- TO_STRING
- TO_TIMESTAMP
- UTCNOW

Fonction DATE_DIFF dans Amazon QLDB

Dans Amazon QLDB, utilisez la`DATE_DIFF` fonction pour renvoyer la différence entre les parties de date spécifiées de deux horodatages donnés.

Syntaxe

```
DATE_DIFF( datetimepart, timestamp1, timestamp2 )
```

Arguments

date/heure (partie)

Références sur lesquelles la fonction opère. Ce paramètre peut avoir l'une des valeurs suivantes :

- `year`
- `month`
- `day`
- `hour`
- `minute`
- `second`

horodatage1, *horodatage2*

Les deux noms de champs ou expressions du type de données`timestamp` que la fonction compare. Si *timestamp2* est postérieur à *timestamp1*, le résultat est positif. Si *timestamp2* est antérieur à *timestamp1*, le résultat est négatif.

Une valeur littérale d'horodatage Ion peut être indiquée par des crochets inverses (`` `...` ``). Pour plus de détails sur le formatage et des exemples de valeurs d'horodatage, consultez la section Horodatages du document de spécifications Amazon Ion.

Type de retour

`int`

Exemples

```
DATE_DIFF(year, `2010-01-01T`, `2011-01-01T`)           -- 1
DATE_DIFF(year, `2010-12T`, `2011-01T`)                 -- 0 (must be at least 12
 months apart to evaluate as a 1 year difference)
DATE_DIFF(month, `2010T`, `2010-05T`)                   -- 4 (2010T is equivalent to
 2010-01-01T00:00:00.000Z)
DATE_DIFF(month, `2010T`, `2011T`)                      -- 12
DATE_DIFF(month, `2011T`, `2010T`)                      -- -12
DATE_DIFF(month, `2010-12-31T`, `2011-01-01T`)          -- 0 (must be at least a full
 month apart to evaluate as a 1 month difference)
DATE_DIFF(day, `2010-01-01T23:00Z`, `2010-01-02T01:00Z`) -- 0 (must be at least 24
 hours apart to evaluate as a 1 day difference)
```

```
-- Runnable statements
SELECT DATE_DIFF(year, `2010-01-01T`, `2011-01-01T`) FROM << 0 >> -- 1
SELECT DATE_DIFF(month, `2010T`, `2010-05T`) FROM << 0 >>         -- 4
```

Références Références

- DATE_ADD
- EXTRACT
- TO_STRING
- TO_TIMESTAMP
- UTCNOW

Fonction EXISTS dans Amazon QLDB

Dans Amazon QLDB pour une valeur PartiQL, utilisez la`EXISTS` fonction pour renvoyer`TRUE` si la valeur est une valeur non vide. Dans le cas contraire, cette fonction renvoie`FALSE`. Si l'entrée de n'`EXISTS`est pas un conteneur, le résultat est`FALSE`.

La`EXISTS` fonction ne se propage pas`NULL` et`MISSING`.

Syntaxe

```
EXISTS ( value )
```

Arguments

`valeur`

Nom ou expression que la fonction évalue. Ce paramètre peut être n'importe lequel des paramètres pris en chargeTypes de données.

Type de retour

`bool`

Exemples

```
EXISTS(`[]`)        -- false (empty list)
```

```
EXISTS(`[1, 2, 3]`) -- true (non-empty list)
EXISTS(`[missing]`) -- true (non-empty list)
EXISTS(`{}`)        -- false (empty struct)
EXISTS(`{ a: 1 }`)  -- true (non-empty struct)
EXISTS(`()`)        -- false (empty s-expression)
EXISTS(`(+ 1 2)`)   -- true (non-empty s-expression)
EXISTS(1)           -- false
EXISTS(`2017T`)     -- false
EXISTS(null)        -- false
EXISTS(missing)     -- error

-- Runnable statements
SELECT EXISTS(`[]`) FROM << 0 >>        -- false
SELECT EXISTS(`[1, 2, 3]`) FROM << 0 >> -- true
```

Références Références

- COALESCE
- NULLIF

Fonction EXTRACT dans Amazon QLDB

Dans Amazon QLDB, utilisez laEXTRACT fonction pour renvoyer la valeur entière d'une date ou d'une partie d'heure spécifiée à partir d'un horodatage donné.

Syntaxe

```
EXTRACT ( datetimepart FROM timestamp )
```

Arguments

date/heure (partie)

Partie de date ou d'heure extraite par la fonction. Ce paramètre peut avoir l'une des valeurs suivantes :

- `year`
- `month`
- `day`
- `hour`

- `minute`
- `second`
- `timezone_hour`
- `timezone_minute`

`timestamp`

Nom du champ ou expression du type de données`timestamp` dont la fonction extrait. Si ce paramètre est de type inconnu (`NULL`ou`MISSING`), la fonction renvoie`NULL`.

Une valeur littérale d'horodatage Ion peut être indiquée par des crochets inverses (`` `...` ``). Pour plus de détails sur le formatage et des exemples de valeurs d'horodatage, consultez la section Horodatages du document de spécifications Amazon Ion.

Type de retour

`int`

Exemples

```
EXTRACT(YEAR FROM `2010-01-01T`)                            -- 2010
EXTRACT(MONTH FROM `2010T`)                                 -- 1 (equivalent to
 2010-01-01T00:00:00.000Z)
EXTRACT(MONTH FROM `2010-10T`)                              -- 10
EXTRACT(HOUR FROM `2017-01-02T03:04:05+07:08`)              -- 3
EXTRACT(MINUTE FROM `2017-01-02T03:04:05+07:08`)            -- 4
EXTRACT(TIMEZONE_HOUR FROM `2017-01-02T03:04:05+07:08`)     -- 7
EXTRACT(TIMEZONE_MINUTE FROM `2017-01-02T03:04:05+07:08`)   -- 8

-- Runnable statements
SELECT EXTRACT(YEAR FROM `2010-01-01T`) FROM << 0 >> -- 2010
SELECT EXTRACT(MONTH FROM `2010T`) FROM << 0 >>      -- 1
```

Références Références

- DATE_ADD
- DATE_DIFF
- TO_STRING
- TO_TIMESTAMP

- UTCNOW

Fonction LOWER dans Amazon QLDB

Dans Amazon QLDB, utilisez la`LOWER` fonction pour convertir tous les caractères majuscules en minuscules dans une chaîne donnée.

Syntaxe

```
LOWER ( string )
```

Arguments

`string`

Nom du champ ou expression du type`string` de données converti par la fonction.

Type de retour

`string`

Exemples

```
SELECT LOWER('AbCdEfG!@#$') FROM << 0 >> -- 'abcdefg!@#$'
```

Références Références

- CHAR_LENGTH
- SUBSTRING
- TRIM
- UPPER

Fonction MAX dans Amazon QLDB

Dans Amazon QLDB, utilisez la`MAX` fonction pour renvoyer la valeur maximale dans un ensemble de valeurs numériques.

Syntaxe

```
MAX ( expression )
```

Arguments

expression

Nom ou expression d'un type de données numérique sur lequel la fonction opère.

Types de données

Types d'arguments pris en charge :

- `int`
- `decimal`
- `float`

Types de retours pris en charge :

- `int`
- `decimal`
- `float`

Exemples

```
SELECT MAX(r.PendingPenaltyTicketAmount) FROM VehicleRegistration r  -- 442.30
SELECT MAX(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>         -- 3
```

Références Références

- AVG
- COUNT
- MIN
- SIZE

- SUM

Fonction MIN dans Amazon QLDB

Dans Amazon QLDB, utilisez la`MIN` fonction pour renvoyer la valeur minimale dans un ensemble de valeurs numériques.

Syntaxe

```
MIN ( expression )
```

Arguments

`expression`

Nom ou expression d'un type de données numérique sur lequel la fonction opère.

Types de données

Types d'arguments pris en charge :

- `int`
- `decimal`
- `float`

Types de retours pris en charge :

- `int`
- `decimal`
- `float`

Exemples

```
SELECT MIN(r.PendingPenaltyTicketAmount) FROM VehicleRegistration r  -- 30.45
SELECT MIN(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>         -- 1
```

Références Références

- AVG
- COUNT
- MAX
- SIZE
- SUM

Fonction NULLIF dans Amazon QLDB

Dans Amazon QLDB pour deux expressions données, utilisez la`NULLIF` fonction pour renvoyer`NULL` si les deux expressions ont la même valeur. Sinon, cette fonction renvoie.

La`NULLIF` fonction ne se propage pas`NULL` et`MISSING`.

Syntaxe

```
NULLIF ( expression1, expression2 )
```

Arguments

`expression1`, *`expression2`*

Les deux noms ou expressions de champ que la fonction compare. Ces paramètres peuvent être n'importe lequel des paramètres pris en chargeTypes de données.

Type de retour

Tout type de données pris en charge. Le type de retour est`NULL` soit identique à celui de la première expression, soit le même.

Exemples

```
NULLIF(1, 1)            -- null
NULLIF(1, 2)            -- 1
NULLIF(1.0, 1)          -- null
```

```
NULLIF(1, '1')           -- 1
NULLIF([1], [1])         -- null
NULLIF(1, NULL)          -- 1
NULLIF(NULL, 1)          -- null
NULLIF(null, null)       -- null
NULLIF(missing, null)    -- null
NULLIF(missing, missing) -- null

-- Runnable statements
SELECT NULLIF(1, 1) FROM << 0 >>   -- null
SELECT NULLIF(1, '1') FROM << 0 >> -- 1
```

Références Références

- COALESCE
- EXISTS

Fonction SIZE dans Amazon QLDB

Dans Amazon QLDB, utilisez la`SIZE` fonction pour renvoyer le nombre d'éléments contenus dans un type de données de conteneur donné (liste, structure ou sac).

Syntaxe

```
SIZE ( container )
```

Arguments

conteneur

Nom ou expression du API sur lequel la fonction opère.

Types de données

Types d'arguments pris en charge :

- list
- structure

- sac

Type de retour :`int`

Si l'entrée de n'`SIZE`est pas un conteneur, la fonction génère une erreur.

Exemples

```
SIZE(`[]`)                  -- 0
SIZE(`[null]`)              -- 1
SIZE(`[1,2,3]`)             -- 3
SIZE(<<'foo', 'bar'>>)      -- 2
SIZE(`{foo: bar}`)          -- 1 (number of key-value pairs)
SIZE(`[{foo: 1}, {foo: 2}]`) -- 2
SIZE(12)                    -- error

-- Runnable statements
SELECT SIZE(`[]`) FROM << 0 >>      -- 0
SELECT SIZE(`[1,2,3]`) FROM << 0 >> -- 3
```

Références Références

- AVG
- COUNT
- MAX
- MIN
- SUM

Fonction SUBSTRING dans Amazon QLDB

Dans Amazon QLDB, utilisez la`SUBSTRING` fonction pour renvoyer une sous-chaîne à partir d'une chaîne donnée. La sous-chaîne commence à partir de l'index de départ spécifié et se termine au dernier caractère de la chaîne, ou à la longueur spécifiée.

Syntaxe

```
SUBSTRING ( string, start-index [, length ] )
```

Arguments

string

Nom du champ ou expression du type de données à`string` partir duquel extraire une sous-chaîne.

index de démarrage

Position de départ dans la *chaîne* à partir de laquelle commencer l'extraction. Ce numéro peut être négatif.

Le premier caractère de la *chaîne* a un indice de 1.

longueur

(Facultatif) Nombre de caractères (points de code) à extraire de la *chaîne*, commençant à l'*index de* départ et se terminant à (*indice* de début et *longueur*) : 1. En d'autres termes, la longueur de la sous-chaîne. Ce numéro ne peut pas être négatif.

Si ce paramètre n'est pas fourni, la fonction continue jusqu'à la fin de la *chaîne*.

Type de retour

`string`

Exemples

```
SUBSTRING('123456789', 0)      -- '123456789'
SUBSTRING('123456789', 1)      -- '123456789'
SUBSTRING('123456789', 2)      -- '23456789'
SUBSTRING('123456789', -4)     -- '123456789'
SUBSTRING('123456789', 0, 999) -- '123456789'
SUBSTRING('123456789', 0, 2)   -- '1'
SUBSTRING('123456789', 1, 999) -- '123456789'
SUBSTRING('123456789', 1, 2)   -- '12'
SUBSTRING('1', 1, 0)           -- ''
SUBSTRING('1', 1, 0)           -- ''
SUBSTRING('1', -4, 0)          -- ''
SUBSTRING('1234', 10, 10)      -- ''

-- Runnable statements
```

```
SELECT SUBSTRING('123456789', 1) FROM << 0 >>    -- "123456789"
SELECT SUBSTRING('123456789', 1, 2) FROM << 0 >> -- "12"
```

Références Références

- CHAR_LENGTH
- LOWER
- TRIM
- UPPER

Fonction SUM dans Amazon QLDB

Dans Amazon QLDB, utilisez la`SUM` fonction pour renvoyer la somme des valeurs du champ de saisie ou de l'expression. Cette fonction fonctionne avec des valeurs numériques et ignore les valeurs nulles ou manquantes.

Syntaxe

```
SUM ( expression )
```

Arguments

`expression`

Nom ou expression d'un type de données numérique sur lequel la fonction opère.

Types de données

Types d'arguments pris en charge :

- `int`
- `decimal`
- `float`

Types de retours pris en charge :

- `int`— Pour les arguments entiers
- `decimal`— Pour les arguments décimaux ou à virgule flottante

Exemples

```
SELECT SUM(r.PendingPenaltyTicketAmount) FROM VehicleRegistration r  -- 735.95
SELECT SUM(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>         -- 6
```

Références Références

- AVG
- COUNT
- MAX
- MIN
- SIZE

Fonction TO_STRING dans Amazon QLDB

Dans Amazon QLDB, utilisez la`TO_STRING` fonction pour renvoyer une représentation sous forme de chaîne d'un horodatage donné dans le modèle de format spécifié.

Syntaxe

```
TO_STRING ( timestamp, 'format' )
```

Arguments

`timestamp`

Nom du champ ou expression du type de données`timestamp` que la fonction convertit en chaîne.

Une valeur littérale d'horodatage Ion peut être indiquée par des crochets inverses (`` `...` ``). Pour plus de détails sur le formatage et des exemples de valeurs d'horodatage, consultez la section Horodatages du document de spécifications Amazon Ion.

format

Chaîne littérale qui spécifie le modèle de format du résultat, en termes de parties datées. Pour connaître les formats valides, consultez Chaînes de format d'horodatage.

Type de retour

`string`

Exemples

```
TO_STRING(`1969-07-20T20:18Z`, 'MMMM d, y')                    -- "July 20, 1969"
TO_STRING(`1969-07-20T20:18Z`, 'MMM d, yyyy')                  -- "Jul 20, 1969"
TO_STRING(`1969-07-20T20:18Z`, 'M-d-yy')                       -- "7-20-69"
TO_STRING(`1969-07-20T20:18Z`, 'MM-d-y')                       -- "07-20-1969"
TO_STRING(`1969-07-20T20:18Z`, 'MMMM d, y h:m a')              -- "July 20, 1969 8:18
 PM"
TO_STRING(`1969-07-20T20:18Z`, 'y-MM-dd''T''H:m:ssX')          --
 "1969-07-20T20:18:00Z"
TO_STRING(`1969-07-20T20:18+08:00Z`, 'y-MM-dd''T''H:m:ssX')    --
 "1969-07-20T20:18:00Z"
TO_STRING(`1969-07-20T20:18+08:00`, 'y-MM-dd''T''H:m:ssXXXX')  --
 "1969-07-20T20:18:00+0800"
TO_STRING(`1969-07-20T20:18+08:00`, 'y-MM-dd''T''H:m:ssXXXXX') --
 "1969-07-20T20:18:00+08:00"

-- Runnable statements
SELECT TO_STRING(`1969-07-20T20:18Z`, 'MMMM d, y') FROM << 0 >>          -- "July 20,
 1969"
SELECT TO_STRING(`1969-07-20T20:18Z`, 'y-MM-dd''T''H:m:ssX') FROM << 0 >> --
 "1969-07-20T20:18:00Z"
```

Références Références

- CAST
- DATE_ADD
- DATE_DIFF
- EXTRACT
- TO_TIMESTAMP
- UTCNOW

Fonction TO_TIMESTAMP dans Amazon QLDB

Dans Amazon QLDB, à partir d'une chaîne représentant un horodatage, utilisez la`TO_TIMESTAMP` fonction pour convertir la chaîne en type de`timestamp` données. Il s'agit de l'opération inverse de`TO_STRING`.

Syntaxe

```
TO_TIMESTAMP ( string [, 'format' ] )
```

Arguments

`string`

Nom du champ ou expression du type de données`string` que la fonction convertit en horodatage.

`format`

(Facultatif) Chaîne littérale qui définit le modèle de format de la *`chaîne`* d'entrée, en termes de parties de date. Pour connaître les formats valides, consultez Chaînes de format d'horodatage.

Si cet argument est omis, la fonction suppose que la *`chaîne`* est au format d'un horodatage Ion standard. C'est la méthode recommandée pour analyser un horodatage Ion à l'aide de cette fonction.

L'ajout de zéro est facultatif lorsque vous utilisez un symbole de format à caractère unique (tel que`y`,`M`,,`d`,`H`,`h`,`s`)`m`,,), mais il est obligatoire pour leurs variantes complétées par des zéros (tels que`yyyy`,`MM`,`dd`,`HH`,`hh`,`mm`,,`ss`).

Un traitement spécial est accordé aux années à deux chiffres (symbole de format`yy`). 1900 est ajoutée aux valeurs supérieures ou égales à 70, et 2000 est ajoutée aux valeurs inférieures à 70.

Les noms des mois et les indicateurs AM ou PM ne distinguent pas les majuscules des minuscules.

Type de retour

`timestamp`

Exemples

```
TO_TIMESTAMP('2007T')                          -- `2007T`
TO_TIMESTAMP('2007-02-23T12:14:33.079-08:00') -- `2007-02-23T12:14:33.079-08:00`
TO_TIMESTAMP('2016', 'y')                      -- `2016T`
TO_TIMESTAMP('2016', 'yyyy')                   -- `2016T`
TO_TIMESTAMP('02-2016', 'MM-yyyy')             -- `2016-02T`
TO_TIMESTAMP('Feb 2016', 'MMM yyyy')           -- `2016-02T`
TO_TIMESTAMP('February 2016', 'MMMM yyyy')     -- `2016-02T`

-- Runnable statements
SELECT TO_TIMESTAMP('2007T') FROM << 0 >>              -- 2007T
SELECT TO_TIMESTAMP('02-2016', 'MM-yyyy') FROM << 0 >> -- 2016-02T
```

Références Références

- CAST
- DATE_ADD
- DATE_DIFF
- EXTRACT
- TO_STRING
- UTCNOW

Fonction TRIM dans Amazon QLDB

Dans Amazon QLDB, utilisez la`TRIM` fonction pour découper une chaîne donnée en supprimant les espaces vides de début et de fin ou un ensemble de caractères spécifié.

Syntaxe

```
TRIM ( [ LEADING | TRAILING | BOTH [ characters ] FROM ] string)
```

Arguments

LEADING

(Facultatif) Indique que les espaces vides ou les caractères spécifiés doivent être supprimés du début de la *chaîne*. S'il n'est pas spécifié, le comportement par défaut est`BOTH`.

TRAILING

(Facultatif) Indique que les espaces vides ou les caractères spécifiés doivent être supprimés de la fin de la *chaîne*. S'il n'est pas spécifié, le comportement par défaut estBOTH.

BOTH

(Facultatif) Indique que les espaces vides de début et de fin ou les caractères spécifiés doivent être supprimés du début et de la fin de la *chaîne*.

caractères

(Facultatif) Le jeu de caractères à supprimer, spécifié sous la forme d'un`string`.

Si ce paramètre n'est pas fourni, les espaces vides sont supprimés.

string

Le nom du champ ou l'expression du type de données`string` que les fonctions suppriment.

Type de retour

`string`

Exemples

```
TRIM('       foobar         ')               -- 'foobar'
TRIM('      \tfoobar\t         ')            -- '\tfoobar\t'
TRIM(LEADING FROM '       foobar         ')  -- 'foobar         '
TRIM(TRAILING FROM '       foobar         ') -- '       foobar'
TRIM(BOTH FROM '       foobar         ')     -- 'foobar'
TRIM(BOTH '1' FROM '11foobar11')             -- 'foobar'
TRIM(BOTH '12' FROM '1112211foobar22211122') -- 'foobar'

-- Runnable statements
SELECT TRIM('       foobar         ') FROM << 0 >>              -- "foobar"
SELECT TRIM(LEADING FROM '       foobar         ') FROM << 0 >> -- "foobar         "
```

Références Références

- CHAR_LENGTH
- LOWER
- SUBSTRING

- UPPER

Fonction TXID dans Amazon QLDB

Dans Amazon QLDB, utilisez la`TXID` fonction pour renvoyer l'identifiant de transaction unique du relevé actuel que vous exécutez. Il s'agit de la valeur attribuée au champ de`txId` métadonnées d'un document lorsque la transaction en cours est validée dans le journal.

Syntaxe

```
TXID()
```

Arguments

Aucune

Type de retour

`string`

Exemples

```
SELECT TXID() FROM << 0 >>  -- "L7S9iJqcn9W2M4qOEn27ay"
SELECT TXID() FROM Person WHERE GovId = 'LEWISR261LL'  -- "BKeMb48PNyvHWJGZHkaodG"
```

Fonction UPPER dans Amazon QLDB

Dans Amazon QLDB, utilisez la`UPPER` fonction pour convertir tous les caractères minuscules en majuscules dans une chaîne donnée.

Syntaxe

```
UPPER ( string )
```

Arguments

`string`

Nom du champ ou expression du type`string` de données converti par la fonction.

Type de retour

`string`

Exemples

```
SELECT UPPER('AbCdEfG!@#$') FROM << 0 >> -- 'ABCDEFG!@#$'
```

Références Références

- CHAR_LENGTH
- LOWER
- SUBSTRING
- TRIM

Fonction UTCNOW dans Amazon QLDB

Dans Amazon QLDB, utilisez la`UTCNOW` fonction pour renvoyer l'heure actuelle en temps universel coordonné (UTC) sous forme de`timestamp`.

Syntaxe

```
UTCNOW()
```

Cette fonction nécessite que vous indiquiez une`FROM` clause dans une`SELECT` requête.

Arguments

Aucune

Type de retour

`timestamp`

Exemples

```
SELECT UTCNOW() FROM << 0 >>  -- 2019-12-27T20:12:16.999Z
SELECT UTCNOW() FROM Person WHERE GovId = 'LEWISR261LL'  -- 2019-12-27T20:12:26.999Z
```

```
INSERT INTO Person VALUE { 'firstName': 'Jane', 'createdAt': UTCNOW() }
UPDATE Person p SET p.updatedAt = UTCNOW() WHERE p.firstName = 'John'
```

Références Références

- DATE_ADD
- DATE_DIFF
- EXTRACT
- TO_STRING
- TO_TIMESTAMP

Chaînes de format d'horodatage

Cette section fournit des informations de référence pour les chaînes de format d'horodatage.

Les chaînes de format d'horodatage s'appliquent aux`TO_TIMESTAMP` fonctions`TO_STRING` et. Ces chaînes peuvent contenir des séparateurs de parties de date (tels que`-` « »,`/` « » ou «`:` ») et les symboles de format suivants.

Format	Exemple	Description
yy	70	Année à deux chiffres
y	1970	Année à quatre chiffres
yyyy	1970	Année sur 4 chiffres avec ajout de zéro
M	1	Entier du mois
MM	01	Entier avec ajout de zéro
MMM	Jan	Nom du mois sous forme abrégée
MMMM	janvier	Nom du mois
d	2	Jour du mois (1—31)

Format	Exemple	Description
dd	02	Jour du mois avec ajout de zéro (de 01 à 31)
a	AM ou PM	Indicateur méridien (pour 12 heures)
h	3	Heure (12 heures, de 1 à 12)
hh	03	Heures avec ajout de zéro (12 heures, de 01 à 12)
H	3	Heure (24 heures, de 0 à 23)
HH	03	Heures avec ajout de zéro (24 heures, de 00 à 23)
m	4	Procès-verbal (0—59)
mm	04	Minutes complétées par des zéros (00—59)
s	5	Secondes (0 à 59)
ss	05	Secondes complétées de zéro (00—59)
S	0	Fraction de seconde (précisio n : 0,1, plage : 0,0-0,9)
SS	06	Fraction de seconde (précisio n : 0,01, plage : 0,0-0,99)
SSS	060	Fraction de seconde (précisio n : 0,001, plage : 0,0-0,999)
X	+07 ou Z	Décalage par rapport à JSON, ou « Z » si le décalage est de 0

Format	Exemple	Description
XX	+0700 ou Z	Décalage par rapport à JSON en heures et minutes, ou « Z » si le décalage est de 0
XXX	+ 07:00 ou Z	Décalage par rapport à JSON en heures et minutes, ou « Z » si le décalage est de 0
x	+07	Décalage par rapport à l'UTC en heures
xx	+700	Décalage par rapport à JSON
xxx	+ 07h00	Décalage par rapport à JSON

Procédstockées stockées stockées stockées stockées stockées stockées stockées stockées stockées stockées stockées stockées stockées QLDB

Dans Amazon QLDB, vous pouvez utiliser laEXEC commande pour exécuter des procédures stockées dans PartiQL selon la syntaxe suivante.

```
EXEC stored_procedure_name argument [, ... ]
```

QLDB uniquement les procédures stockées QLDS DS :

Rubriques

- Procédstockées RED

Procédstockées RED

> **Note**
>
> Tous les registres créés avant le 22 juillet 2021 ne sont actuellement pas éligibles à la rédaction. Vous pouvez consulter l'heure de création de votre registre sur la console Amazon QLDB.

Dans Amazon QLDB, utilisez la procédure`REDACT_REVISION` enregistrée pour supprimer définitivement une révision de document individuelle et inactive à la fois dans le stockage indexé et dans le stockage du journal. Cette procédure stockée supprime toutes les données utilisateur de la révision spécifiée. Toutefois, la séquence du journal et les métadonnées du document, y compris l'identifiant et le hachage du document, restent inchangées. Cette opération est irréversible.

La révision du document spécifiée doit être une révision inactive dans l'historique. La dernière révision active d'un document n'est pas éligible à la rédaction.

Après avoir soumis une demande de rédaction en exécutant cette procédure stockée, QLDB traite la rédaction des données de manière asynchrone. Une fois la rédaction terminée, les données utilisateur de la révision spécifiée (représentées par la`data` structure) sont remplacées par un nouveau`dataHash` champ. La valeur de ce champ est le hachage Amazon Ion de la`data` structure supprimée. Par conséquent, le registre conserve l'intégrité globale de ses données et reste vérifiable cryptographiquement par le biais des opérations d'API de vérification existantes.

Pour un exemple d'opération de rédaction avec des exemples de données, voirExemple de rédaction dansRédaction de révisions de documents.

> **Note**
>
> Pour savoir comment contrôler l'accès afin d'exécuter cette commande PartiQL sur des tables spécifiques, consultezCommencer à utiliser le mode d'autorisation standard dans Amazon QLDB.

Rubriques

Considérations et restrictions relatives à la rédaction et restrictions relatives

Avant de commencer à rédiger des données dans Amazon QLDB, assurez-vous de prendre connaissance des considérations et limites suivantes :

- La procédure`REDACT_REVISION` stockée cible vos données utilisateur dans le cadre d'une révision de document individuelle et inactive. Pour supprimer plusieurs révisions, vous devez exécuter la procédure enregistrée une fois pour chaque révision. Vous pouvez supprimer une révision par transaction.
- Pour supprimer des champs particuliers dans une révision de document, vous devez d'abord utiliser une instruction DML (Data Manipulation Language) distincte pour modifier la révision. Pour plus d'informations, veuillez consulter Supprimer un champ particulier dans une révision.
- Une fois que QLDB a reçu une demande de rédaction, vous ne pouvez ni annuler ni modifier la demande. Pour vérifier si une rédaction est complète, vous pouvez vérifier si la`data` structure d'une révision a été remplacée par un`dataHash` champ. Pour en savoir plus, consultez Vérifier si une rédaction est complète.
- La rédaction n'a aucun impact sur les données QLDB répliquées en dehors du service QLDB. Cela inclut toutes les exportations Amazon S3 Kinesis Data StreDS. Vous devez utiliser d'autres méthodes de conservation des données pour gérer toutes les données stockées en dehors de QLDB.
- La rédaction n'a aucun impact sur les valeurs littérales des instructions PartiQL enregistrées dans le journal. Il est recommandé d'réaliser des instructions stockées de la variable plutôt que des valeurs stockées pour réaliser leur travail. Un espace réservé est écrit dans le journal sous la forme d'un point d'interrogation (?) au lieu de toute information sensible susceptible de nécessiter une rédaction.

  Pour savoir comment exécuter des instructions PartiQL par programmation à l'aide du pilote QLDB, consultez les didacticiels correspondant à chaque langage de programmation pris en charge dansDémarrage.

Syntaxe

```
EXEC REDACT_REVISION `block-address`, 'table-id', 'document-id'
```

Arguments

`*adresse de bloc*`

Emplacement du bloc de journal de la révision du document à rédiger. Une adresse est une structure Amazon Ion qui comporte deux champs :`strandId` et`sequenceNo`.

Il s'agit d'une valeur littérale d'ions qui est indiquée par des crochets inverses. Par exemple :

```
`{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`
```

Pour savoir comment trouver l'adresse de blocage, consultezInterroger les métadonnées d'un document.

« *ID de table* »

L'ID unique du tableau dont vous souhaitez modifier la version du document, indiqué par des guillemets simples.

Pour savoir comment trouver l'ID de table, consultezInterrogation du catalogue système.

« *identifiant du document* »

L'ID de document unique de la révision à rédiger, indiqué par des guillemets simples.

Pour savoir comment trouver l'identifiant du document, consultezInterroger les métadonnées d'un document.

Valeur renvoyée

Une structure Amazon Ion qui représente la révision du document à rédiger, au format suivant.

```
{
  blockAddress: {
    strandId: String,
    sequenceNo: Int
```

```
  },
  tableId: String,
  documentId: String,
  version: Int
}
```

Champs de structure de retour

- `blockAddress`— Emplacement du bloc de journal de la révision à rédiger. Une adresse comporte les deux champs suivants.
  - `strandId`— L'identifiant unique du volet du journal qui contient le bloc.
  - `sequenceNo`— Numéro d'index qui indique l'emplacement du bloc dans le brin.
- `tableId`— L'identifiant unique de la table dont vous êtes en train de modifier la version.
- `documentId`— L'identifiant de document unique de la révision à rédiger.
- `version`— Le numéro de version de la révision du document à rédiger.

Voici un exemple de structure de retour avec des exemples de données.

```
{
  blockAddress: {
    strandId: "CsRnx0RDoNK6ANEEePa1ov",
    sequenceNo: 134
  },
  tableId: "6GZumdHggklLdMGyQq9DNX",
  documentId: "IXlQPSbfyKMIIsygePeKrZ",
  version: 0
}
```

Exemples

```
EXEC REDACT_REVISION `{strandId:"7z2P0AyQKWD8oFYmGNhi8D", sequenceNo:7}`,
 '8F0TPCmdNQ6JTRpiLj2TmW', '05K8zpGYWynDlEOK5afDRc'
```

Opérateurs partiQL dans Amazon QLDB

PartiQL dans Amazon QLDB prend en charge les opérateurs standard SQL suivants.

> **Note**
>
> Les opérateurs SQL qui ne figurent pas dans cette liste ne sont actuellement pas pris en charge dans QLDB.

Opérateurs arithmétiques

Opérateur	Description
+	Addition
-	Soustraction
*	Multiplication
/	Division
%	Module

Opérateurs de comparaison

Opérateur	Description
=	Egal à
>	Supérieure à
<	Inférieur à
>=	Supérieur ou égal à
<=	Inférieur ou égal à
<>	Non égal à

Opérateurs logiques

Opérateur	Description
`AND`	`TRUE` si toutes les conditions séparées par `AND` sont `TRUE`
`BETWEEN`	`TRUE` si l'opérande est comprise dans la plage des comparaisons
`IN`	`TRUE` si l'opérande est égal à l'une d'une liste d'expressions
`IS`	`TRUE`si l'opérande est un type de données donné, y compris`NULL` ou`MISSING`
`LIKE`	`TRUE`si l'opérande correspond à un modèle
`NOT`	Inverse la valeur d'une expression booléenne donnée
`OR`	`TRUE`si l'une des conditions séparées par`OR` est`TRUE`

Opérateurs de chaîne

Opérateur	Description
`||`	Concatène deux chaînes de chaque côté de l'`||`opérateur et renvoie la chaîne concaténé e. Si une chaîne ou les deux ont la valeur null`NULL`, le résultat de la concaténation est null.

Mots clés réservés dans Amazon QLDB

La liste suivante recense les mots réservés PartiQL dans Amazon QLDB. Vous pouvez utiliser un mot clé réservé comme identifiant entre guillemets doubles (par exemple,`"user"`). Pour plus d'informations sur les conventions de citation de PartiQL dans QLDB, consultezInterroger Ion avec PartiQL.

> **Important**
>
> Les mots clés de cette liste sont tous considérés comme réservés car PartiQL est rétrocompatible avec SQL-92. Cependant, QLDB ne prend en charge qu'un sous-ensemble de ces mots réservés. Pour obtenir la liste des mots clés SQL actuellement QLDB charge, veuillez consulter les rubriques suivantes :
>
> - Références Références Références de PartiQL
> - Opérateurs PartiQL
> - Commandes PartiQL

```
ABSOLUTE
ACTION
ADD
ALL
ALLOCATE
ALTER
AND
ANY
ARE
AS
ASC
ASSERTION
AT
AUTHORIZATION
AVG
BAG
BEGIN
BETWEEN
BIT
BIT_LENGTH
BLOB
```

```
BOOL
BOOLEAN
BOTH
BY
CASCADE
CASCADED
CASE
CAST
CATALOG
CHAR
CHARACTER
CHARACTER_LENGTH
CHAR_LENGTH
CHECK
CLOB
CLOSE
COALESCE
COLLATE
COLLATION
COLUMN
COMMIT
CONNECT
CONNECTION
CONSTRAINT
CONSTRAINTS
CONTINUE
CONVERT
CORRESPONDING
COUNT
CREATE
CROSS
CURRENT
CURRENT_DATE
CURRENT_TIME
CURRENT_TIMESTAMP
CURRENT_USER
CURSOR
DATE
DATE_ADD
DATE_DIFF
DAY
DEALLOCATE
DEC
DECIMAL
```

```
DECLARE
DEFAULT
DEFERRABLE
DEFERRED
DELETE
DESC
DESCRIBE
DESCRIPTOR
DIAGNOSTICS
DISCONNECT
DISTINCT
DOMAIN
DOUBLE
DROP
ELSE
END
END-EXEC
ESCAPE
EXCEPT
EXCEPTION
EXEC
EXECUTE
EXISTS
EXTERNAL
EXTRACT
FALSE
FETCH
FIRST
FLOAT
FOR
FOREIGN
FOUND
FROM
FULL
GET
GLOBAL
GO
GOTO
GRANT
GROUP
HAVING
HOUR
IDENTITY
IMMEDIATE
```

```
IN
INDEX
INDICATOR
INITIALLY
INNER
INPUT
INSENSITIVE
INSERT
INT
INTEGER
INTERSECT
INTERVAL
INTO
IS
ISOLATION
JOIN
KEY
LANGUAGE
LAST
LEADING
LEFT
LEVEL
LIKE
LIMIT
LIST
LOCAL
LOWER
MATCH
MAX
MIN
MINUTE
MISSING
MODULE
MONTH
NAMES
NATIONAL
NATURAL
NCHAR
NEXT
NO
NOT
NULL
NULLIF
NUMERIC
```

```
OCTET_LENGTH
OF
ON
ONLY
OPEN
OPTION
OR
ORDER
OUTER
OUTPUT
OVERLAPS
PAD
PARTIAL
PIVOT
POSITION
PRECISION
PREPARE
PRESERVE
PRIMARY
PRIOR
PRIVILEGES
PROCEDURE
PUBLIC
READ
REAL
REFERENCES
RELATIVE
REMOVE
RESTRICT
REVOKE
RIGHT
ROLLBACK
ROWS
SCHEMA
SCROLL
SECOND
SECTION
SELECT
SESSION
SESSION_USER
SET
SEXP
SIZE
SMALLINT
```

```
SOME
SPACE
SQL
SQLCODE
SQLERROR
SQLSTATE
STRING
STRUCT
SUBSTRING
SUM
SYMBOL
SYSTEM_USER
TABLE
TEMPORARY
THEN
TIME
TIMESTAMP
TIMEZONE_HOUR
TIMEZONE_MINUTE
TO
TO_STRING
TO_TIMESTAMP
TRAILING
TRANSACTION
TRANSLATE
TRANSLATION
TRIM
TRUE
TUPLE
TXID
UNDROP
UNION
UNIQUE
UNKNOWN
UNPIVOT
UPDATE
UPPER
USAGE
USER
USING
UTCNOW
VALUE
VALUES
VARCHAR
```

```
VARYING
VIEW
WHEN
WHENEVER
WHERE
WITH
WORK
WRITE
YEAR
ZONE
```

Référence du format de données Amazon Ion dans Amazon QLDB

Amazon QLDB utilise un modèle de notation de données qui unifie Amazon Ion avec un sous-ensemble de types PartiQL. Cette section fournit une présentation de référence du format de données du document Ion, indépendamment de son intégration à PartiQL.

Interroger Ion avec PartiQL dans Amazon QLDB

Pour connaître la syntaxe et la sémantique de l'interrogation de données Ion à l'aide de PartiQL dans QLDB, consultezInterroger Ion avec PartiQL la référence Amazon QLDB PartiQL.

Pour des exemples de code qui interrogent et traitent des données Ion dans un registre QLDB, consultezExemples de code Amazon Ion etUtilisation d'Amazon Ion.

Rubriques

- Qu'est-ce qu'Amazon Ion
- Spécification Ion
- Compatible avec JSON
- Extensions depuis JSON
- Exemple de texte Ion
- Références d'API
- Exemples de code Amazon Ion dans QLDB

Qu'est-ce qu'Amazon Ion

Ion est un format de sérialisation de données hiérarchique open source, richement typé et autodescriptif qui a été initialement développé en interne chez Amazon. Il est basé sur un modèle de

données abstrait qui vous permet de stocker des données structurées et non structurées. Il s'agit d'un sur-ensemble de JSON, ce qui signifie que tout document JSON valide est également un document Ion valide. Ce guide suppose une connaissance pratique de base de JSON. Si vous ne connaissez pas encore JSON, consultez Présentation du JSON pour plus d'informations.

Vous pouvez noter les documents Ion de manière interchangeable sous forme de texte lisible par l'homme ou sous forme codée en binaire. Comme le JSON, le formulaire texte est facile à lire et à écrire, ce qui permet un prototypage rapide. Le codage binaire est plus compact et plus efficace pour la persistance, la transmission et l'analyse. Un processeur ionique peut transcoder entre les deux formats pour représenter exactement le même ensemble de structures de données sans perte de données. Cette fonctionnalité permet aux applications d'optimiser la façon dont elles traitent les données pour différents cas d'utilisation.

> **Note**
>
> Le modèle de données Ion est strictement basé sur des valeurs et ne prend pas en charge les références. Ainsi, le modèle de données peut représenter des hiérarchies de données qui peuvent être imbriquées à une profondeur arbitraire, mais pas des graphes orientés.

Spécification Ion

Pour obtenir la liste complète des types de données Ion Core avec des descriptions complètes et des informations détaillées sur le formatage des valeurs, consultez le document de spécification Ion sur le GitHub site Amazon.

Pour rationaliser le développement d'applications, Amazon Ion fournit des bibliothèques clientes qui traitent les données Ion pour vous. Pour des exemples de code illustrant des cas d'utilisation courants pour le traitement de données Ion, consultez le livre de recettes Amazon Ion sur GitHub.

Compatible avec JSON

Comme au format JSON, vous composez des documents Amazon Ion avec un ensemble de types de données primitifs et un ensemble de types de conteneurs définis de manière récursive. Ion inclut les types de données JSON traditionnels suivants :

- `null`: valeur nulle (vide) générique non typée. De plus, comme décrit dans la section suivante, Ion prend en charge un type nul distinct pour chaque type primitif.

- `bool`: valeurs booléennes.
- `string`: littéraux de texte Unicode.
- `list`: ensembles de valeurs hétérogènes ordonnés.
- `struct`: collections non ordonnées de paires nom-valeur. Comme JSON, il`struct` autorise plusieurs valeurs par nom, mais cela est généralement déconseillé.

Extensions depuis JSON

Types de numéros

Au lieu du`number` type JSON ambigu, Amazon Ion définit strictement les nombres comme l'un des types suivants :

- `int`: entiers signés de taille arbitraire.
- `decimal`: nombres réels codés en décimales d'une précision arbitraire.
- `float`: nombres à virgule flottante codés en binaire (IEEE 64 bits).

Lors de l'analyse des documents, un processeur ionique attribue les types de numéros suivants :

- `int`: nombres sans exposant ni point décimal (par exemple,`100200`).
- `decimal`: nombres avec virgule décimale et sans exposant (par exemple`0.00001`,`200.0`).
- `float`: nombres avec un exposant, tels que la notation scientifique ou la notation E (par exemple,`2e0`,`3.1e-4`).

Nouveaux types de données

Amazon Ion ajoute les types de données suivants :

- `timestamp`: moments de date/heure/fuseau horaire d'une précision arbitraire.
- `symbol`: atomes symboliques Unicode (tels que les identifiants).
- `blob`: données binaires encodées par l'utilisateur.
- `clob`: données de texte dont le codage est défini par l'utilisateur.
- `sexp`: collections de valeurs ordonnées avec une sémantique définie par l'application.

Types nuls

Outre le type nul générique défini par JSON, Amazon Ion prend en charge un type nul distinct pour chaque type primitif. Cela indique un manque de valeur tout en conservant un type de données strict.

```
null
null.null       // Identical to untyped null
null.bool
null.int
null.float
null.decimal
null.timestamp
null.string
null.symbol
null.blob
null.clob
null.struct
null.list
null.sexp
```

Exemple de texte Ion

```
// Here is a struct, which is similar to a JSON object.
{
    // Field names don't always have to be quoted.
    name: "fido",

    // This is an integer.
    age: 7,

    // This is a timestamp with day precision.
    birthday: 2012-03-01T,

    // Here is a list, which is like a JSON array.
    toys: [
        // These are symbol values, which are like strings,
        // but get encoded as integers in binary.
        ball,
        rope
    ],
}
```

Références d'API

- ion-go
- ion-java
- ion-js
- ion-python

Exemples de code Amazon Ion dans QLDB

Cette section fournit des exemples de code qui traitent les données Amazon Ion en lisant et en écrivant des valeurs de documents dans un registre Amazon QLDB. Les exemples de code utilisent le pilote QLDB pour exécuter des instructions PartiQL sur le registre. Ces exemples font partie de l'exemple d'application dansDémarrer avec Amazon QLDB à l'aide d'un exemple de didacticiel d'application et sont open source sur le GitHub siteAWS Samples.

Pour des exemples de code généraux illustrant les cas d'utilisation courants du traitement de données Ion, consultez le livre de recettes Amazon Ion sur GitHub.

Exécution du code

Le code du didacticiel pour chaque langage de programmation effectue les étapes suivantes :

1. Connect au registre`vehicle-registration` d'échantillons.
2. Créez une table nommée`IonTypes`.
3. Insérez un document dans le tableau avec un seul`Name` champ.
4. Pour chaque type de données Ion pris en charge :
   a. Mettez à jour le`Name` champ du document avec une valeur littérale du type de données.
   b. Recherchez la table pour obtenir la dernière révision du document.
   c. Vérifiez que la valeur de`Name` a conservé ses propriétés de type de données d'origine en vérifiant qu'elle correspond au type attendu.
5. Oubliez la`IonTypes` table.

> **Note**
>
> Avant d'exécuter ce code de didacticiel, vous devez créer un registre nommé`vehicle-registration`.

Java

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazon.ion.IonBlob;
import com.amazon.ion.IonBool;
import com.amazon.ion.IonClob;
import com.amazon.ion.IonDecimal;
import com.amazon.ion.IonFloat;
import com.amazon.ion.IonInt;
import com.amazon.ion.IonList;
import com.amazon.ion.IonNull;
import com.amazon.ion.IonSexp;
```

```
import com.amazon.ion.IonString;
import com.amazon.ion.IonStruct;
import com.amazon.ion.IonSymbol;
import com.amazon.ion.IonTimestamp;
import com.amazon.ion.IonValue;
import com.amazon.ion.Timestamp;
import com.fasterxml.jackson.annotation.JsonCreator;
import com.fasterxml.jackson.annotation.JsonProperty;
import java.util.Collections;
import java.util.List;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;

/**
 * Insert all the supported Ion types into a ledger and verify that they are stored
 and can be retrieved properly, retaining
 * their original properties.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-credentials.html
 */
public class InsertIonTypes {
    public static final Logger log = LoggerFactory.getLogger(InsertIonTypes.class);
    public static final String TABLE_NAME = "IonTypes";

    private InsertIonTypes() {}

    /**
     * Update a document's Name value in the database. Then, query the value of the
 Name key and verify the expected Ion type was
     * saved.
     *
     * @param txn
     *              The {@link TransactionExecutor} for statement execution.
     * @param ionValue
     *              The {@link IonValue} to set the document's Name value to.
     *
     * @throws AssertionError when no value is returned for the Name key or if the
 value does not match the expected type.
     */
    public static void updateRecordAndVerifyType(final TransactionExecutor txn,
 final IonValue ionValue) {
```

```
        final String updateStatement = String.format("UPDATE %s SET Name = ?",
TABLE_NAME);
        final List<IonValue> parameters = Collections.singletonList(ionValue);
        txn.execute(updateStatement, parameters);
        log.info("Updated document.");

        final String searchQuery = String.format("SELECT VALUE Name FROM %s",
TABLE_NAME);
        final Result result = txn.execute(searchQuery);

        if (result.isEmpty()) {
            throw new AssertionError("Did not find any values for the Name key.");
        }
        for (IonValue value : result) {
            if (!ionValue.getClass().isInstance(value)) {
                throw new AssertionError(String.format("The queried value, %s, is
not an instance of %s.",
                        value.getClass().toString(),
ionValue.getClass().toString()));
            }
            if (!value.getType().equals(ionValue.getType())) {
                throw new AssertionError(String.format("The queried value type, %s,
does not match %s.",
                        value.getType().toString(), ionValue.getType().toString()));
            }
        }

        log.info("Successfully verified value is instance of {} with type {}.",
ionValue.getClass().toString(),
                ionValue.getType().toString());
    }

    /**
     * Delete a table.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param tableName
     *              The name of the table to delete.
     */
    public static void deleteTable(final TransactionExecutor txn, final String
tableName) {
        log.info("Deleting {} table...", tableName);
        final String statement = String.format("DROP TABLE %s", tableName);
```

```
        txn.execute(statement);
        log.info("{} table successfully deleted.", tableName);
    }

    public static void main(final String... args) {
        final IonBlob ionBlob = Constants.SYSTEM.newBlob("hello".getBytes());
        final IonBool ionBool = Constants.SYSTEM.newBool(true);
        final IonClob ionClob = Constants.SYSTEM.newClob("{{'This is a CLOB of
text.'}}".getBytes());
        final IonDecimal ionDecimal = Constants.SYSTEM.newDecimal(0.1);
        final IonFloat ionFloat = Constants.SYSTEM.newFloat(0.2);
        final IonInt ionInt = Constants.SYSTEM.newInt(1);
        final IonList ionList = Constants.SYSTEM.newList(new int[]{1, 2});
        final IonNull ionNull = Constants.SYSTEM.newNull();
        final IonSexp ionSexp = Constants.SYSTEM.newSexp(new int[]{2, 3});
        final IonString ionString = Constants.SYSTEM.newString("string");
        final IonStruct ionStruct = Constants.SYSTEM.newEmptyStruct();
        ionStruct.put("brand", Constants.SYSTEM.newString("ford"));
        final IonSymbol ionSymbol = Constants.SYSTEM.newSymbol("abc");
        final IonTimestamp ionTimestamp =
Constants.SYSTEM.newTimestamp(Timestamp.now());

        final IonBlob ionNullBlob = Constants.SYSTEM.newNullBlob();
        final IonBool ionNullBool = Constants.SYSTEM.newNullBool();
        final IonClob ionNullClob = Constants.SYSTEM.newNullClob();
        final IonDecimal ionNullDecimal = Constants.SYSTEM.newNullDecimal();
        final IonFloat ionNullFloat = Constants.SYSTEM.newNullFloat();
        final IonInt ionNullInt = Constants.SYSTEM.newNullInt();
        final IonList ionNullList = Constants.SYSTEM.newNullList();
        final IonSexp ionNullSexp = Constants.SYSTEM.newNullSexp();
        final IonString ionNullString = Constants.SYSTEM.newNullString();
        final IonStruct ionNullStruct = Constants.SYSTEM.newNullStruct();
        final IonSymbol ionNullSymbol = Constants.SYSTEM.newNullSymbol();
        final IonTimestamp ionNullTimestamp = Constants.SYSTEM.newNullTimestamp();


        ConnectToLedger.getDriver().execute(txn -> {
            CreateTable.createTable(txn, TABLE_NAME);
            final Document document = new
Document(Constants.SYSTEM.newString("val"));
            InsertDocument.insertDocuments(txn, TABLE_NAME,
Collections.singletonList(document));

            updateRecordAndVerifyType(txn, ionBlob);
```

```
            updateRecordAndVerifyType(txn, ionBool);
            updateRecordAndVerifyType(txn, ionClob);
            updateRecordAndVerifyType(txn, ionDecimal);
            updateRecordAndVerifyType(txn, ionFloat);
            updateRecordAndVerifyType(txn, ionInt);
            updateRecordAndVerifyType(txn, ionList);
            updateRecordAndVerifyType(txn, ionNull);
            updateRecordAndVerifyType(txn, ionSexp);
            updateRecordAndVerifyType(txn, ionString);
            updateRecordAndVerifyType(txn, ionStruct);
            updateRecordAndVerifyType(txn, ionSymbol);
            updateRecordAndVerifyType(txn, ionTimestamp);

            updateRecordAndVerifyType(txn, ionNullBlob);
            updateRecordAndVerifyType(txn, ionNullBool);
            updateRecordAndVerifyType(txn, ionNullClob);
            updateRecordAndVerifyType(txn, ionNullDecimal);
            updateRecordAndVerifyType(txn, ionNullFloat);
            updateRecordAndVerifyType(txn, ionNullInt);
            updateRecordAndVerifyType(txn, ionNullList);
            updateRecordAndVerifyType(txn, ionNullSexp);
            updateRecordAndVerifyType(txn, ionNullString);
            updateRecordAndVerifyType(txn, ionNullStruct);
            updateRecordAndVerifyType(txn, ionNullSymbol);
            updateRecordAndVerifyType(txn, ionNullTimestamp);

            deleteTable(txn, TABLE_NAME);
        });
    }

    /**
     * This class represents a simple document with a single key, Name, to use for
the IonTypes table.
     */
    private static class Document {
        private final IonValue name;

        @JsonCreator
        private Document(@JsonProperty("Name") final IonValue name) {
            this.name = name;
        }

        @JsonProperty("Name")
        private IonValue getName() {
```

```
            return name;
        }
    }
}
```

Node.js

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QldbDriver, Result, TransactionExecutor } from "amazon-qldb-driver-nodejs";
import { AssertionError } from "assert";
import { dom, IonType, IonTypes } from "ion-js";

import { insertDocument } from "./InsertDocument";
import { getQldbDriver } from "./ConnectToLedger";
import { createTable } from "./CreateTable";
import { error, log } from "./qldb/LogUtil";

const TABLE_NAME: string = "IonTypes";

/**
```

```
 * Delete a table.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param tableName Name of the table to delete.
 * @returns Promise which fulfills with void.
 */
export async function deleteTable(txn: TransactionExecutor, tableName: string):
 Promise<void> {
    log(`Deleting ${tableName} table...`);
    const statement: string = `DROP TABLE ${tableName}`;
    await txn.execute(statement);
    log(`${tableName} table successfully deleted.`);
}

/**
 * Update a document's Name value in QLDB. Then, query the value of the Name key and
 verify the expected Ion type was
 * saved.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param parameter The IonValue to set the document's Name value to.
 * @param ionType The Ion type that the Name value should be.
 * @returns Promise which fulfills with void.
 */
async function updateRecordAndVerifyType(
    txn: TransactionExecutor,
    parameter: any,
    ionType: IonType
): Promise<void> {
    const updateStatement: string = `UPDATE ${TABLE_NAME} SET Name = ?`;
    await txn.execute(updateStatement, parameter);
    log("Updated record.");

    const searchStatement: string = `SELECT VALUE Name FROM ${TABLE_NAME}`;
    const result: Result = await txn.execute(searchStatement);

    const results: dom.Value[] = result.getResultList();

    if (0 === results.length) {
        throw new AssertionError({
            message: "Did not find any values for the Name key."
        });
    }

    results.forEach((value: dom.Value) => {
        if (value.getType().binaryTypeId !== ionType.binaryTypeId) {
```

```
            throw new AssertionError({
                message: `The queried value type, ${value.getType().name}, does not
 match expected type, ${ionType.name}.`
            });
        }
    });

    log(`Successfully verified value is of type ${ionType.name}.`);
}

/**
 * Insert all the supported Ion types into a table and verify that they are stored
 and can be retrieved properly,
 * retaining their original properties.
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
    try {
        const qldbDriver: QldbDriver = getQldbDriver();
        await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
            await createTable(txn, TABLE_NAME);
            await insertDocument(txn, TABLE_NAME, [{ "Name": "val" }]);
            await updateRecordAndVerifyType(txn, dom.load("null"), IonTypes.NULL);
            await updateRecordAndVerifyType(txn, true, IonTypes.BOOL);
            await updateRecordAndVerifyType(txn, 1, IonTypes.INT);
            await updateRecordAndVerifyType(txn, 3.2, IonTypes.FLOAT);
            await updateRecordAndVerifyType(txn, dom.load("5.5"), IonTypes.DECIMAL);
            await updateRecordAndVerifyType(txn, dom.load("2020-02-02"),
 IonTypes.TIMESTAMP);
            await updateRecordAndVerifyType(txn, dom.load("abc123"),
 IonTypes.SYMBOL);
            await updateRecordAndVerifyType(txn, dom.load("\"string\""),
 IonTypes.STRING);
            await updateRecordAndVerifyType(txn, dom.load("{{ \"clob\" }}"),
 IonTypes.CLOB);
            await updateRecordAndVerifyType(txn, dom.load("{{ blob }}"),
 IonTypes.BLOB);
            await updateRecordAndVerifyType(txn, dom.load("(1 2 3)"),
 IonTypes.SEXP);
            await updateRecordAndVerifyType(txn, dom.load("[1, 2, 3]"),
 IonTypes.LIST);
            await updateRecordAndVerifyType(txn, dom.load("{brand: ford}"),
 IonTypes.STRUCT);
            await deleteTable(txn, TABLE_NAME);
```

```
        });
    } catch (e) {
        error(`Error updating and validating Ion types: ${e}`);
    }
}

if (require.main === module) {
    main();
}
```

Python

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy, modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from datetime import datetime
from decimal import Decimal
from logging import basicConfig, getLogger, INFO

from amazon.ion.simple_types import IonPyBool, IonPyBytes, IonPyDecimal, IonPyDict,
 IonPyFloat, IonPyInt, IonPyList, \
    IonPyNull, IonPySymbol, IonPyText, IonPyTimestamp
from amazon.ion.simpleion import loads
from amazon.ion.symbols import SymbolToken
from amazon.ion.core import IonType
```

```
from pyqldbsamples.create_table import create_table
from pyqldbsamples.constants import Constants
from pyqldbsamples.insert_document import insert_documents
from pyqldbsamples.model.sample_data import convert_object_to_ion
from pyqldbsamples.connect_to_ledger import create_qldb_driver

logger = getLogger(__name__)
basicConfig(level=INFO)

TABLE_NAME = 'IonTypes'


def update_record_and_verify_type(driver, parameter, ion_object, ion_type):
    """
    Update a record in the database table. Then query the value of the record and
 verify correct ion type saved.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type parameter: :py:class:`amazon.ion.simple_types.IonPyValue`
    :param parameter: The Ion value or Python native type that is convertible to Ion
 for filling in parameters of the
                      statement.

    :type
 ion_object: :py:obj:`IonPyBool`/:py:obj:`IonPyBytes`/:py:obj:`IonPyDecimal`/:py:obj:`IonPyD

 /:py:obj:`IonPyFloat`/:py:obj:`IonPyInt`/:py:obj:`IonPyList`/:py:obj:`IonPyNull`

 /:py:obj:`IonPySymbol`/:py:obj:`IonPyText`/:py:obj:`IonPyTimestamp`
    :param ion_object: The Ion object to verify against.

    :type ion_type: :py:class:`amazon.ion.core.IonType`
    :param ion_type: The Ion type to verify against.

    :raises TypeError: When queried value is not an instance of Ion type.
    """
    update_query = 'UPDATE {} SET Name = ?'.format(TABLE_NAME)
    driver.execute_lambda(lambda executor: executor.execute_statement(update_query,
 parameter))
    logger.info('Updated record.')
```

```
    search_query = 'SELECT VALUE Name FROM {}'.format(TABLE_NAME)
    cursor = driver.execute_lambda(lambda executor:
 executor.execute_statement(search_query))

    for c in cursor:
        if not isinstance(c, ion_object):
            raise TypeError('The queried value is not an instance of
 {}'.format(ion_object.__name__))

        if c.ion_type is not ion_type:
            raise TypeError('The queried value type does not match
 {}'.format(ion_type))

    logger.info("Successfully verified value is instance of '{}' with type
 '{}'.".format(ion_object.__name__, ion_type))
    return cursor


def delete_table(driver, table_name):
    """
    Delete a table.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type table_name: str
    :param table_name: Name of the table to delete.

    :rtype: int
    :return: The number of changes to the database.
    """
    logger.info("Deleting '{}' table...".format(table_name))
    cursor = driver.execute_lambda(lambda executor: executor.execute_statement('DROP
 TABLE {}'.format(table_name)))
    logger.info("'{}' table successfully deleted.".format(table_name))
    return len(list(cursor))


def insert_and_verify_ion_types(driver):
    """
    Insert all the supported Ion types and Python values that are convertible to Ion
 into a ledger and verify that they
    are stored and can be retrieved properly, retaining their original properties.
```

```
    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: A QLDB Driver object.
    """
    python_bytes = str.encode('hello')
    python_bool = True
    python_float = float('0.2')
    python_decimal = Decimal('0.1')
    python_string = "string"
    python_int = 1
    python_null = None
    python_datetime = datetime(2016, 12, 20, 5, 23, 43)
    python_list = [1, 2]
    python_dict = {"brand": "Ford"}

    ion_clob = convert_object_to_ion(loads('{{"This is a CLOB of text."}}'))
    ion_blob = convert_object_to_ion(python_bytes)
    ion_bool = convert_object_to_ion(python_bool)
    ion_decimal = convert_object_to_ion(python_decimal)
    ion_float = convert_object_to_ion(python_float)
    ion_int = convert_object_to_ion(python_int)
    ion_list = convert_object_to_ion(python_list)
    ion_null = convert_object_to_ion(python_null)
    ion_sexp = convert_object_to_ion(loads('(cons 1 2)'))
    ion_string = convert_object_to_ion(python_string)
    ion_struct = convert_object_to_ion(python_dict)
    ion_symbol = convert_object_to_ion(SymbolToken(text='abc', sid=123))
    ion_timestamp = convert_object_to_ion(python_datetime)

    ion_null_clob = convert_object_to_ion(loads('null.clob'))
    ion_null_blob = convert_object_to_ion(loads('null.blob'))
    ion_null_bool = convert_object_to_ion(loads('null.bool'))
    ion_null_decimal = convert_object_to_ion(loads('null.decimal'))
    ion_null_float = convert_object_to_ion(loads('null.float'))
    ion_null_int = convert_object_to_ion(loads('null.int'))
    ion_null_list = convert_object_to_ion(loads('null.list'))
    ion_null_sexp = convert_object_to_ion(loads('null.sexp'))
    ion_null_string = convert_object_to_ion(loads('null.string'))
    ion_null_struct = convert_object_to_ion(loads('null.struct'))
    ion_null_symbol = convert_object_to_ion(loads('null.symbol'))
    ion_null_timestamp = convert_object_to_ion(loads('null.timestamp'))

    create_table(driver, TABLE_NAME)
    insert_documents(driver, TABLE_NAME, [{'Name': 'val'}])
    update_record_and_verify_type(driver, python_bytes, IonPyBytes, IonType.BLOB)
```

```
    update_record_and_verify_type(driver, python_bool, IonPyBool, IonType.BOOL)
    update_record_and_verify_type(driver, python_float, IonPyFloat, IonType.FLOAT)
    update_record_and_verify_type(driver, python_decimal, IonPyDecimal,
IonType.DECIMAL)
    update_record_and_verify_type(driver, python_string, IonPyText, IonType.STRING)
    update_record_and_verify_type(driver, python_int, IonPyInt, IonType.INT)
    update_record_and_verify_type(driver, python_null, IonPyNull, IonType.NULL)
    update_record_and_verify_type(driver, python_datetime, IonPyTimestamp,
IonType.TIMESTAMP)
    update_record_and_verify_type(driver, python_list, IonPyList, IonType.LIST)
    update_record_and_verify_type(driver, python_dict, IonPyDict, IonType.STRUCT)
    update_record_and_verify_type(driver, ion_clob, IonPyBytes, IonType.CLOB)
    update_record_and_verify_type(driver, ion_blob, IonPyBytes, IonType.BLOB)
    update_record_and_verify_type(driver, ion_bool, IonPyBool, IonType.BOOL)
    update_record_and_verify_type(driver, ion_decimal, IonPyDecimal,
IonType.DECIMAL)
    update_record_and_verify_type(driver, ion_float, IonPyFloat, IonType.FLOAT)
    update_record_and_verify_type(driver, ion_int, IonPyInt, IonType.INT)
    update_record_and_verify_type(driver, ion_list, IonPyList, IonType.LIST)
    update_record_and_verify_type(driver, ion_null, IonPyNull, IonType.NULL)
    update_record_and_verify_type(driver, ion_sexp, IonPyList, IonType.SEXP)
    update_record_and_verify_type(driver, ion_string, IonPyText, IonType.STRING)
    update_record_and_verify_type(driver, ion_struct, IonPyDict, IonType.STRUCT)
    update_record_and_verify_type(driver, ion_symbol, IonPySymbol, IonType.SYMBOL)
    update_record_and_verify_type(driver, ion_timestamp, IonPyTimestamp,
IonType.TIMESTAMP)
    update_record_and_verify_type(driver, ion_null_clob, IonPyNull, IonType.CLOB)
    update_record_and_verify_type(driver, ion_null_blob, IonPyNull, IonType.BLOB)
    update_record_and_verify_type(driver, ion_null_bool, IonPyNull, IonType.BOOL)
    update_record_and_verify_type(driver, ion_null_decimal, IonPyNull,
IonType.DECIMAL)
    update_record_and_verify_type(driver, ion_null_float, IonPyNull, IonType.FLOAT)
    update_record_and_verify_type(driver, ion_null_int, IonPyNull, IonType.INT)
    update_record_and_verify_type(driver, ion_null_list, IonPyNull, IonType.LIST)
    update_record_and_verify_type(driver, ion_null_sexp, IonPyNull, IonType.SEXP)
    update_record_and_verify_type(driver, ion_null_string, IonPyNull,
IonType.STRING)
    update_record_and_verify_type(driver, ion_null_struct, IonPyNull,
IonType.STRUCT)
    update_record_and_verify_type(driver, ion_null_symbol, IonPyNull,
IonType.SYMBOL)
    update_record_and_verify_type(driver, ion_null_timestamp, IonPyNull,
IonType.TIMESTAMP)
    delete_table(driver, TABLE_NAME)
```

```
def main(ledger_name=Constants.LEDGER_NAME):
    """
    Insert all the supported Ion types and Python values that are convertible to Ion
 into a ledger and verify that they
    are stored and can be retrieved properly, retaining their original properties.
    """
    try:
        with create_qldb_driver(ledger_name) as driver:
            insert_and_verify_ion_types(driver)
    except Exception as e:
        logger.exception('Error updating and validating Ion types.')
        raise e


if __name__ == '__main__':
    main()
```

Référence d'API Amazon QLDB

Ce chapitre décrit les opérations d'API de bas niveau pour Amazon QLDB qui sont accessibles via HTTP, leAWS Command Line Interface(AWS CLI), ou unAWSKIT SDK :

- Amazon QLDB— L'API de gestion des ressources QLDB (également connue sous le nom deplan de contrôle). Cette API est utilisée uniquement pour la gestion des ressources du registre et pour les opérations de données non transactionnelles. Vous pouvez utiliser ces opérations pour créer, supprimer, décrire, répertorier et mettre à jour des registres. Vous pouvez également vérifier les données des journaux par cryptographie et exporter ou diffuser des blocs de journaux.
- Session Amazon QLDB— L'API de données transactionnelles QLDB. Vous pouvez utiliser cette API pour exécuter des transactions de données sur un livre avecPartiQLInstructions

> **Important**
>
> Au lieu d'interagir directement avec leSession QLDBAPI, nous recommandons d'utiliser le pilote QLDB ou le shell QLDB pour exécuter des transactions de données sur un livre.
>
> - Si vous travaillez avec unAWSSDK, utilisez le pilote QLDB. Le pilote fournit une couche d'abstraction de haut niveau au-dessus deSession QLDBdata et gère le`SendCommand`opération pour vous. Pour plus d'informations et pour obtenir la liste des langages de programmation pris en chargeDémarrage.
> - Si vous travaillez avec leAWS CLI, utilisez le shell QLDB. L'interpréteur de commandes est une interface de ligne de commande qui utilise le pilote QLDB pour interagir avec un registre. Pour plus d'informations, consultez Utilisation du shell Amazon QLDB (API de données uniquement).

Rubriques

Actions

Les actions suivantes sont prises en charge par le service Amazon QLDB :

- CancelJournalKinesisStream
- CreateLedger
- DeleteLedger
- DescribeJournalKinesisStream
- DescribeJournalS3Export
- DescribeLedger
- ExportJournalToS3
- GetBlock
- GetDigest
- GetRevision
- ListJournalKinesisStreamsForLedger
- ListJournalS3Exports
- ListJournalS3ExportsForLedger
- ListLedgers
- ListTagsForResource
- StreamJournalToKinesis
- TagResource
- UntagResource
- UpdateLedger
- UpdateLedgerPermissionsMode

Les actions suivantes sont prises en charge par la session Amazon QLDB

- SendCommand

Amazon QLDB

The following actions are supported by Amazon QLDB:

- CancelJournalKinesisStream
- CreateLedger
- DeleteLedger
- DescribeJournalKinesisStream
- DescribeJournalS3Export
- DescribeLedger
- ExportJournalToS3
- GetBlock
- GetDigest
- GetRevision
- ListJournalKinesisStreamsForLedger
- ListJournalS3Exports
- ListJournalS3ExportsForLedger
- ListLedgers
- ListTagsForResource
- StreamJournalToKinesis
- TagResource
- UntagResource
- UpdateLedger
- UpdateLedgerPermissionsMode

CancelJournalKinesisStream

Service : Amazon QLDB

Termine un flux de journal Amazon QLDB donné. Avant qu'un stream puisse être annulé, son statut actuel doit êtreACTIVE.

Vous ne pouvez pas redémarrer un stream après l'avoir annulé. Les ressources de flux QLDB annulées sont soumises à une période de rétention de 7 jours. Elles sont donc automatiquement supprimées après l'expiration de cette limite.

Syntaxe de la demande

```
DELETE /ledgers/name/journal-kinesis-streams/streamId HTTP/1.1
```

Paramètres de demande URI

La demande utilise les paramètres URI suivants.

name

Nom du registre.

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Modèle : `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatoire : oui

streamId

L'UUID (représenté dans le texte codé en Base62) du flux de journal QLDB à annuler.

Contraintes de longueur : longueur fixe de 22.

Modèle : `^[A-Za-z-0-9]+$`

Obligatoire : oui

Corps de la demande

La demande n’a pas de corps de requête.

Syntaxe de la réponse

```
HTTP/1.1 200
Content-type: application/json

{
   "StreamId": "string"
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées au format JSON par le service.

StreamId

L'UUID (texte codé en Base62) du flux de journal QLDB annulé.

Type : chaîne

Contraintes de longueur : longueur fixe de 22.

Modèle : `^[A-Za-z-0-9]+$`

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

InvalidParameterException

Un ou plusieurs paramètres de la demande ne sont pas valides.

Code d'état HTTP : 400

ResourceNotFoundException

La ressource spécifiée n'existe pas.

Code d'état HTTP : 404

ResourcePreconditionNotMetException

L'opération a échoué car une condition n'était pas satisfaite à l'avance.

Code d'état HTTP : 412

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

CreateLedger

Service : Amazon QLDB

Crée un nouveau registre Compte AWS dans votre région actuelle.

Syntaxe de la demande

```
POST /ledgers HTTP/1.1
Content-type: application/json

{
   "DeletionProtection": boolean,
   "KmsKey": "string",
   "Name": "string",
   "PermissionsMode": "string",
   "Tags": {
      "string" : "string"
   }
}
```

Paramètres de demande URI

La demande n'utilise pas de paramètres URI.

Corps de la demande

Cette demande accepte les données suivantes au format JSON.

DeletionProtection

Spécifie si le registre est protégé contre la suppression par un utilisateur. À défaut de définition lors de la création du registre, cette fonctionnalité est activée (`true`) par défaut.

Si la protection contre la suppression est activée, vous devez commencer par la désactiver avant de pouvoir supprimer le registre. Vous pouvez la désactiver en appelant l'opération `UpdateLedger` pour définir ce paramètre sur `false`.

Type : booléen

Obligatoire : non

KmsKey

La clé in AWS Key Management Service (AWS KMS) à utiliser pour le chiffrement des données au repos dans le registre. Pour plus d'informations, veuillez consulter la rubrique Chiffrement au repos dans le Guide du développeur Amazon QLDB.

Utilisez l'une des options suivantes pour spécifier ce paramètre :

- `AWS_OWNED_KMS_KEY`: utilisez une AWS KMS clé détenue et gérée par AWS vous.
- Non défini : par défaut, utilisez une clé KMS AWS détenue.
- Une clé KMS symétrique gérée par le client valide : utilisez la clé KMS de chiffrement symétrique spécifiée dans votre compte que vous créez, possédez et gérez.

  Amazon QLDB ne prend pas en charge les clés asymétriques. Pour plus d'informations, consultez la section Utilisation de clés symétriques et asymétriques dans le manuel du AWS Key Management Service développeur.

Pour spécifier une clé KMS gérée par le client, utilisez son ID de clé, son Amazon Resource Name (ARN), son nom d'alias ou son ARN d'alias. Lorsque vous utilisez un nom d'alias, préfixez-le avec `"alias/"`. Pour spécifier une clé dans un autre Compte AWS, vous devez utiliser l'ARN de la clé ou l'alias ARN.

Par exemple :

- ID de clé : `1234abcd-12ab-34cd-56ef-1234567890ab`
- ARN de clé : `arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`
- Nom d'alias : `alias/ExampleAlias`
- ARN d'alias : `arn:aws:kms:us-east-2:111122223333:alias/ExampleAlias`

Pour plus d'informations, consultez la section Identifiants clés (KeyId) dans le guide du AWS Key Management Service développeur.

Type : chaîne

Contraintes de longueur : longueur maximale de 1600.

Obligatoire : non

Name

Nom du registre que vous souhaitez créer. Le nom doit être unique parmi tous les registres de votre Compte AWS région actuelle.

Les contraintes de dénomination pour les noms de registre sont définies dans Quotas dans Amazon QLDB du Amazon QLDB Developer Guide.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Modèle : `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatoire : oui

PermissionsMode

Mode d'autorisations à attribuer au registre que vous souhaitez créer. Ce paramètre peut avoir une des valeurs suivantes :

- `ALLOW_ALL` : un mode d'autorisations hérité qui permet le contrôle d'accès avec une granularité au niveau de l'API pour les registres.

  Ce mode permet aux utilisateurs qui possèdent l'autorisation d'API `SendCommand` pour ce registre d'exécuter toutes les commandes PartiQL (par conséquent, `ALLOW_ALL`) sur toutes les tables du registre spécifié. Ce mode ignore les politiques d'autorisations IAM au niveau de la table ou de la commande que vous créez pour le registre.
- `STANDARD` : (Recommandé) un mode d'autorisations qui permet le contrôle d'accès avec une granularité plus fine pour les registres, les tables et les commandes PartiQL.

  Par défaut, ce mode refuse toutes les demandes utilisateur d'exécuter des commandes PartiQL sur les tables de ce registre. Pour autoriser l'exécution des commandes PartiQL, vous devez créer des politiques d'autorisations IAM pour des ressources de table spécifiques et des actions PartiQL, en plus de l'autorisation d'API `SendCommand` pour le registre. Pour plus d'informations, veuillez consulter la rubrique Premiers pas avec le mode d'autorisations standard dans le Guide du développeur Amazon QLDB.

> **Note**
>
> Nous vous recommandons vivement d'utiliser le mode d'autorisations `STANDARD` pour optimiser la sécurité des données de votre registre.

Type : chaîne

Valeurs valides : `ALLOW_ALL | STANDARD`

Obligatoire : oui

Tags

Les paires clé-valeur à ajouter sous forme de balises au registre que vous souhaitez créer. Les clés de balises sont sensibles à la casse. Les valeurs des balises distinguent les majuscules et minuscules et peuvent être nulles.

Type : mappage chaîne/chaîne

Entrées cartographiques : nombre minimum de 0 éléments. Nombre maximum de 200 éléments.

Contraintes de longueur de clé : longueur minimale de 1. Longueur maximale de 128.

Contraintes de longueur de valeur : longueur minimale de 0. Longueur maximale de 256.

Obligatoire : non

Syntaxe de la réponse

```
HTTP/1.1 200
Content-type: application/json

{
   "Arn": "string",
   "CreationDateTime": number,
   "DeletionProtection": boolean,
   "KmsKeyArn": "string",
   "Name": "string",
   "PermissionsMode": "string",
   "State": "string"
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées au format JSON par le service.

Arn

Le nom de ressource Amazon (ARN) du registre.

Type : chaîne

Contraintes de longueur : longueur minimale de 20. Longueur maximale de 1600.

CreationDateTime

Date et heure, au format epoch time, auxquelles le registre a été créé. (Le format Epoch est le nombre de secondes écoulées depuis 00h00 le 1er janvier 1970 UTC.)

Type : Timestamp

DeletionProtection

Spécifie si le registre est protégé contre la suppression par un utilisateur. À défaut de définition lors de la création du registre, cette fonctionnalité est activée (`true`) par défaut.

Si la protection contre la suppression est activée, vous devez commencer par la désactiver avant de pouvoir supprimer le registre. Vous pouvez la désactiver en appelant l'opération `UpdateLedger` pour définir ce paramètre sur `false`.

Type : booléen

KmsKeyArn

L'ARN de la clé KMS gérée par le client que le registre utilise pour le chiffrement au repos. Si ce paramètre n'est pas défini, le registre utilise une clé KMS AWS détenue pour le chiffrement.

Type : chaîne

Contraintes de longueur : longueur minimale de 20. Longueur maximale de 1600.

Name

Nom du registre.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Modèle : `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

PermissionsMode

Mode d'autorisations du registre que vous avez créé.

Type : chaîne

Valeurs valides : `ALLOW_ALL | STANDARD`

State

État actuel du registre.

Type : chaîne

Valeurs valides : `CREATING | ACTIVE | DELETING | DELETED`

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

InvalidParameterException

Un ou plusieurs paramètres de la demande ne sont pas valides.

Code d’état HTTP : 400

LimitExceededException

Vous avez atteint la limite du nombre maximum de ressources autorisées.

Code d’état HTTP : 400

ResourceAlreadyExistsException

La ressource spécifiée existe déjà.

Code d’état HTTP : 409

ResourceInUseException

La ressource spécifiée ne peut pas être modifiée pour le moment.

Code d’état HTTP : 409

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

DeleteLedger

Service : Amazon QLDB

Supprime un registre et l'ensemble de son contenu. Cette action est irréversible.

Si la protection contre la suppression est activée, vous devez commencer par la désactiver avant de pouvoir supprimer le registre. Vous pouvez la désactiver en appelant l'opération `UpdateLedger` pour définir ce paramètre sur `false`.

Syntaxe de la demande

```
DELETE /ledgers/name HTTP/1.1
```

Paramètres de demande URI

La demande utilise les paramètres URI suivants.

name

Nom du registre que vous souhaitez supprimer.

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Modèle : `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatoire : oui

Corps de la demande

La demande n’a pas de corps de requête.

Syntaxe de la réponse

```
HTTP/1.1 200
```

Eléments de réponse

Si l’action aboutit, le service renvoie une réponse HTTP 200 avec un corps HTTP vide.

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

InvalidParameterException

Un ou plusieurs paramètres de la demande ne sont pas valides.

Code d'état HTTP : 400

ResourceInUseException

La ressource spécifiée ne peut pas être modifiée pour le moment.

Code d'état HTTP : 409

ResourceNotFoundException

La ressource spécifiée n'existe pas.

Code d'état HTTP : 404

ResourcePreconditionNotMetException

L'opération a échoué car une condition n'était pas satisfaite à l'avance.

Code d'état HTTP : 412

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

DescribeJournalKinesisStream

Service : Amazon QLDB

Renvoie des informations détaillées sur un flux de journal Amazon QLDB donné. La sortie inclut le nom de ressource Amazon (ARN), le nom du flux, l'état actuel, l'heure de création et les paramètres de la demande de création de flux d'origine.

Cette action ne renvoie aucun flux de journal expiré. Pour plus d'informations, consultez la section Expiration des flux de terminaux dans le manuel Amazon QLDB Developer Guide.

Syntaxe de la demande

```
GET /ledgers/name/journal-kinesis-streams/streamId HTTP/1.1
```

Paramètres de demande URI

La demande utilise les paramètres URI suivants.

name

Nom du registre.

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Modèle : `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatoire : oui

streamId

L'UUID (représenté dans un texte codé en Base62) du flux de journal QLDB à décrire.

Contraintes de longueur : longueur fixe de 22.

Modèle : `^[A-Za-z-0-9]+$`

Obligatoire : oui

Corps de la demande

La demande n’a pas de corps de requête.

Syntaxe de la réponse

```
HTTP/1.1 200
Content-type: application/json

{
   "Stream": {
      "Arn": "string",
      "CreationTime": number,
      "ErrorCause": "string",
      "ExclusiveEndTime": number,
      "InclusiveStartTime": number,
      "KinesisConfiguration": {
         "AggregationEnabled": boolean,
         "StreamArn": "string"
      },
      "LedgerName": "string",
      "RoleArn": "string",
      "Status": "string",
      "StreamId": "string",
      "StreamName": "string"
   }
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées au format JSON par le service.

Stream

Informations relatives au flux de journal QLDB renvoyé par une requête. `DescribeJournalS3Export`

Type : objet JournalKinesisStreamDescription

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

InvalidParameterException

Un ou plusieurs paramètres de la demande ne sont pas valides.

Code d'état HTTP : 400

ResourceNotFoundException

La ressource spécifiée n'existe pas.

Code d'état HTTP : 404

ResourcePreconditionNotMetException

L'opération a échoué car une condition n'était pas satisfaite à l'avance.

Code d'état HTTP : 412

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

DescribeJournalS3Export

Service : Amazon QLDB

Renvoie des informations sur une tâche d'exportation de journal, notamment le nom du registre, l'ID d'exportation, l'heure de création, le statut actuel et les paramètres de la demande de création d'exportation d'origine.

Cette action ne renvoie aucune tâche d'exportation expirée. Pour plus d'informations, consultez Exporter l'expiration des tâches dans le manuel Amazon QLDB Developer Guide.

Si la tâche d'exportation avec le paramètre donné `ExportId` n'existe pas, elle est lancée`ResourceNotFoundException`.

Si le registre contenant le donné `Name` n'existe pas, il est lancé. `ResourceNotFoundException`

Syntaxe de la demande

```
GET /ledgers/name/journal-s3-exports/exportId HTTP/1.1
```

Paramètres de demande URI

La demande utilise les paramètres URI suivants.

exportId

L'UUID (représenté dans du texte codé en Base62) de la tâche d'exportation du journal à décrire.

Contraintes de longueur : longueur fixe de 22.

Modèle : `^[A-Za-z-0-9]+$`

Obligatoire : oui

name

Nom du registre.

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Modèle : `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatoire : oui

Corps de la demande

La demande n'a pas de corps de requête.

Syntaxe de la réponse

```
HTTP/1.1 200
Content-type: application/json

{
   "ExportDescription": {
      "ExclusiveEndTime": number,
      "ExportCreationTime": number,
      "ExportId": "string",
      "InclusiveStartTime": number,
      "LedgerName": "string",
      "OutputFormat": "string",
      "RoleArn": "string",
      "S3ExportConfiguration": {
         "Bucket": "string",
         "EncryptionConfiguration": {
            "KmsKeyArn": "string",
            "ObjectEncryptionType": "string"
         },
         "Prefix": "string"
      },
      "Status": "string"
   }
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées au format JSON par le service.

ExportDescription

Informations relatives à la tâche d'exportation du journal renvoyées par une `DescribeJournalS3Export` demande.

Type : objet JournalS3ExportDescription

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

ResourceNotFoundException

La ressource spécifiée n’existe pas.

Code d’état HTTP : 404

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

DescribeLedger

Service : Amazon QLDB

Renvoie des informations sur un registre, notamment son état, son mode d'autorisation, les paramètres de chiffrement au repos et sa date de création.

Syntaxe de la demande

```
GET /ledgers/name HTTP/1.1
```

Paramètres de demande URI

La demande utilise les paramètres URI suivants.

name

Nom du registre que vous souhaitez décrire.

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Modèle : `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatoire : oui

Corps de la demande

La demande n'a pas de corps de requête.

Syntaxe de la réponse

```
HTTP/1.1 200
Content-type: application/json

{
   "Arn": "string",
   "CreationDateTime": number,
   "DeletionProtection": boolean,
   "EncryptionDescription": {
      "EncryptionStatus": "string",
      "InaccessibleKmsKeyDateTime": number,
      "KmsKeyArn": "string"
```

```
    },
    "Name": "string",
    "PermissionsMode": "string",
    "State": "string"
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées au format JSON par le service.

Arn

Le nom de ressource Amazon (ARN) du registre.

Type : chaîne

Contraintes de longueur : longueur minimale de 20. Longueur maximale de 1600.

CreationDateTime

Date et heure, au format epoch time, auxquelles le registre a été créé. (Le format Epoch est le nombre de secondes écoulées depuis 00h00 le 1er janvier 1970 UTC.)

Type : Timestamp

DeletionProtection

Spécifie si le registre est protégé contre la suppression par un utilisateur. À défaut de définition lors de la création du registre, cette fonctionnalité est activée (`true`) par défaut.

Si la protection contre la suppression est activée, vous devez commencer par la désactiver avant de pouvoir supprimer le registre. Vous pouvez la désactiver en appelant l'opération `UpdateLedger` pour définir ce paramètre sur `false`.

Type : booléen

EncryptionDescription

Informations sur le chiffrement des données au repos dans le registre. Cela inclut l'état actuel, la AWS KMS clé et le moment où la clé est devenue inaccessible (en cas d'erreur). Si ce paramètre n'est pas défini, le registre utilise une clé KMS AWS détenue pour le chiffrement.

Type : objet LedgerEncryptionDescription

Name

Nom du registre.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Modèle : `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

PermissionsMode

Le mode d'autorisations du registre.

Type : chaîne

Valeurs valides : `ALLOW_ALL | STANDARD`

State

État actuel du registre.

Type : chaîne

Valeurs valides : `CREATING | ACTIVE | DELETING | DELETED`

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

InvalidParameterException

Un ou plusieurs paramètres de la demande ne sont pas valides.

Code d’état HTTP : 400

ResourceNotFoundException

La ressource spécifiée n’existe pas.

Code d’état HTTP : 404

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

ExportJournalToS3

Service : Amazon QLDB

Exporte le contenu d'un journal dans une plage de dates et d'heures depuis un registre vers un compartiment Amazon Simple Storage Service (Amazon S3) spécifié. Une tâche d'exportation de journal peut écrire les objets de données sous forme de texte ou de représentation binaire au format Amazon Ion, ou au format de texte JSON Lines.

Si le registre contenant le donné `Name` n'existe pas, il est lancé. `ResourceNotFoundException`

Si le registre contenant le donné `Name` est en `CREATING` état, il est lancé. `ResourcePreconditionNotMetException`

Vous pouvez lancer jusqu'à deux demandes d'exportation de journaux simultanées pour chaque registre. Au-delà de cette limite, les demandes d'exportation de journaux sont émises`LimitExceededException`.

Syntaxe de la demande

```
POST /ledgers/name/journal-s3-exports HTTP/1.1
Content-type: application/json

{
   "ExclusiveEndTime": number,
   "InclusiveStartTime": number,
   "OutputFormat": "string",
   "RoleArn": "string",
   "S3ExportConfiguration": {
      "Bucket": "string",
      "EncryptionConfiguration": {
         "KmsKeyArn": "string",
         "ObjectEncryptionType": "string"
      },
      "Prefix": "string"
   }
}
```

Paramètres de demande URI

La demande utilise les paramètres URI suivants.

name

Nom du registre.

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Modèle : `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatoire : oui

Corps de la demande

Cette demande accepte les données suivantes au format JSON.

ExclusiveEndTime

Date et heure de fin exclusives de la gamme de contenus de journaux à exporter.

Le paramètre `ExclusiveEndTime` doit être au format date et heure `ISO 8601` et en temps universel coordonné (UTC). Par exemple : `2019-06-13T21:36:34Z`.

La valeur `ExclusiveEndTime` doit être inférieure ou égale à la date et à l'heure UTC actuelles.

Type : Timestamp

Obligatoire : oui

InclusiveStartTime

Date et heure de début incluses pour la gamme de contenus du journal à exporter.

Le paramètre `InclusiveStartTime` doit être au format date et heure `ISO 8601` et en temps universel coordonné (UTC). Par exemple : `2019-06-13T21:36:34Z`.

Ça `InclusiveStartTime` doit être avant`ExclusiveEndTime`.

Si vous fournissez un `InclusiveStartTime` qui se trouve avant celui du registre`CreationDateTime`, Amazon QLDB le définit par défaut sur celui du registre. `CreationDateTime`

Type : Timestamp

Obligatoire : oui

OutputFormat

Format de sortie des données de journal exportées. Une tâche d'exportation de journal peut écrire les objets de données sous forme de texte ou de représentation binaire au format Amazon Ion, ou au format de texte JSON Lines.

Par défaut : `ION_TEXT`

Au format JSON Lines, chaque bloc de journal d'un objet de données exporté est un objet JSON valide délimité par une nouvelle ligne. Vous pouvez utiliser ce format pour intégrer directement les exportations JSON à des outils d'analyse tels qu'Amazon Athena et AWS Glue parce que ces services peuvent analyser automatiquement le JSON délimité par de nouvelles lignes.

Type : chaîne

Valeurs valides : `ION_BINARY | ION_TEXT | JSON`

Obligatoire : non

RoleArn

Le nom de ressource Amazon (ARN) du rôle IAM qui accorde les autorisations QLDB pour une tâche d'exportation de journal permet d'effectuer les opérations suivantes :

- Écrivez des objets dans votre compartiment Amazon S3.
- (Facultatif) Utilisez votre clé gérée par le client dans AWS Key Management Service (AWS KMS) pour le chiffrement côté serveur de vos données exportées.

Pour transmettre un rôle à QLDB lors d'une demande d'exportation de journal, vous devez disposer des autorisations nécessaires pour effectuer `iam:PassRole` l'action sur la ressource du rôle IAM. Cela est obligatoire pour toutes les demandes d'exportation de journaux.

Type : chaîne

Contraintes de longueur : longueur minimale de 20. Longueur maximale de 1600.

Obligatoire : oui

S3ExportConfiguration

Les paramètres de configuration de la destination du compartiment Amazon S3 pour votre demande d'exportation.

Type : objet S3ExportConfiguration

Obligatoire : oui

Syntaxe de la réponse

```
HTTP/1.1 200
Content-type: application/json

{
   "ExportId": "string"
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées au format JSON par le service.

ExportId

L'UUID (représenté dans du texte codé en Base62) que QLDB attribue à chaque tâche d'exportation de journal.

Pour décrire votre demande d'exportation et vérifier l'état de la tâche, vous pouvez utiliser `ExportId` to call`DescribeJournalS3Export`.

Type : chaîne

Contraintes de longueur : longueur fixe de 22.

Modèle : `^[A-Za-z-0-9]+$`

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

ResourceNotFoundException

La ressource spécifiée n'existe pas.

Code d'état HTTP : 404

ResourcePreconditionNotMetException

L'opération a échoué car une condition n'était pas satisfaite à l'avance.

Code d’état HTTP : 412

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

GetBlock

Service : Amazon QLDB

Renvoie un objet de type bloc à une adresse spécifiée dans un journal. Renvoie également une preuve du bloc spécifié pour vérification si elle `DigestTipAddress` est fournie.

Pour plus d'informations sur le contenu des données d'un bloc, consultez le contenu du journal dans le manuel Amazon QLDB Developer Guide.

Si le registre spécifié n'existe pas ou est en `DELETING` état, il est lancé. `ResourceNotFoundException`

Si le registre spécifié est en `CREATING` état, il lance`ResourcePreconditionNotMetException`.

Si aucun bloc n'existe avec l'adresse spécifiée, il est lancé`InvalidParameterException`.

Syntaxe de la demande

```
POST /ledgers/name/block HTTP/1.1
Content-type: application/json

{
   "BlockAddress": {
      "IonText": "string"
   },
   "DigestTipAddress": {
      "IonText": "string"
   }
}
```

Paramètres de demande URI

La demande utilise les paramètres URI suivants.

name

Nom du registre.

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Modèle : `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatoire : oui

Corps de la demande

Cette demande accepte les données suivantes au format JSON.

BlockAddress

L'emplacement du bloc que vous souhaitez demander. Une adresse est une structure Amazon Ion qui comporte deux champs : `strandId` et`sequenceNo`.

Par exemple : `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:14}`.

Type : objet ValueHolder

Obligatoire : oui

DigestTipAddress

Le dernier emplacement du bloc couvert par le résumé pour lequel une preuve doit être demandée. Une adresse est une structure Amazon Ion qui comporte deux champs : `strandId` et`sequenceNo`.

Par exemple : `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:49}`.

Type : objet ValueHolder

Obligatoire : non

Syntaxe de la réponse

```
HTTP/1.1 200
Content-type: application/json

{
   "Block": {
      "IonText": "string"
   },
   "Proof": {
      "IonText": "string"
   }
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées au format JSON par le service.

Block

L'objet de données de bloc au format Amazon Ion.

Type : objet ValueHolder

Proof

L'objet de preuve au format Amazon Ion renvoyé par une `GetBlock` demande. Une preuve contient la liste des valeurs de hachage requises pour recalculer le condensé spécifié à l'aide d'un arbre Merkle, en commençant par le bloc spécifié.

Type : objet ValueHolder

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

InvalidParameterException

Un ou plusieurs paramètres de la demande ne sont pas valides.

Code d’état HTTP : 400

ResourceNotFoundException

La ressource spécifiée n’existe pas.

Code d’état HTTP : 404

ResourcePreconditionNotMetException

L'opération a échoué car une condition n'était pas satisfaite à l'avance.

Code d’état HTTP : 412

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

GetDigest

Service : Amazon QLDB

Renvoie le résumé d'un registre contenant le dernier bloc validé dans le journal. La réponse inclut une valeur de hachage de 256 bits et une adresse de bloc.

Syntaxe de la demande

```
POST /ledgers/name/digest HTTP/1.1
```

Paramètres de demande URI

La demande utilise les paramètres URI suivants.

name

Nom du registre.

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Modèle : `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatoire : oui

Corps de la demande

La demande n’a pas de corps de requête.

Syntaxe de la réponse

```
HTTP/1.1 200
Content-type: application/json

{
   "Digest": blob,
   "DigestTipAddress": {
      "IonText": "string"
   }
}
```

Eléments de réponse

Si l’action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées au format JSON par le service.

Digest

La valeur de hachage de 256 bits représentant le condensé renvoyé par une demande. `GetDigest`

Type : objet de données binaires encodées en base64

Contraintes de longueur : longueur fixe de 32.

DigestTipAddress

L'emplacement du dernier bloc couvert par le résumé que vous avez demandé. Une adresse est une structure Amazon Ion qui comporte deux champs : `strandId` et`sequenceNo`.

Type : objet ValueHolder

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

InvalidParameterException

Un ou plusieurs paramètres de la demande ne sont pas valides.

Code d’état HTTP : 400

ResourceNotFoundException

La ressource spécifiée n’existe pas.

Code d’état HTTP : 404

ResourcePreconditionNotMetException

L'opération a échoué car une condition n'était pas satisfaite à l'avance.

Code d’état HTTP : 412

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

GetRevision

Service : Amazon QLDB

Renvoie un objet de données de révision pour un ID de document et une adresse de bloc spécifiés. Renvoie également une preuve de la révision spécifiée pour vérification si elle `DigestTipAddress` est fournie.

Syntaxe de la demande

```
POST /ledgers/name/revision HTTP/1.1
Content-type: application/json

{
   "BlockAddress": {
      "IonText": "string"
   },
   "DigestTipAddress": {
      "IonText": "string"
   },
   "DocumentId": "string"
}
```

Paramètres de demande URI

La demande utilise les paramètres URI suivants.

name

Nom du registre.

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Modèle : `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatoire : oui

Corps de la demande

Cette demande accepte les données suivantes au format JSON.

BlockAddress

Emplacement du bloc où se trouve la révision du document à vérifier. Une adresse est une structure Amazon Ion qui comporte deux champs : `strandId` et`sequenceNo`.

Par exemple : `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:14}`.

Type : objet ValueHolder

Obligatoire : oui

DigestTipAddress

Le dernier emplacement du bloc couvert par le résumé pour lequel une preuve doit être demandée. Une adresse est une structure Amazon Ion qui comporte deux champs : `strandId` et`sequenceNo`.

Par exemple : `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:49}`.

Type : objet ValueHolder

Obligatoire : non

DocumentId

L'UUID (représenté dans le texte codé en Base62) du document à vérifier.

Type : chaîne

Contraintes de longueur : longueur fixe de 22.

Modèle : `^[A-Za-z-0-9]+$`

Obligatoire : oui

Syntaxe de la réponse

```
HTTP/1.1 200
Content-type: application/json

{
   "Proof": {
      "IonText": "string"
   },
```

```
   "Revision": {
      "IonText": "string"
   }
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées au format JSON par le service.

Proof

L'objet de preuve au format Amazon Ion renvoyé par une `GetRevision` demande. Une preuve contient la liste des valeurs de hachage nécessaires pour recalculer le condensé spécifié à l'aide d'un arbre Merkle, en commençant par la révision du document spécifiée.

Type : objet ValueHolder

Revision

L'objet de données de révision du document au format Amazon Ion.

Type : objet ValueHolder

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

InvalidParameterException

Un ou plusieurs paramètres de la demande ne sont pas valides.

Code d'état HTTP : 400

ResourceNotFoundException

La ressource spécifiée n'existe pas.

Code d'état HTTP : 404

ResourcePreconditionNotMetException

L'opération a échoué car une condition n'était pas satisfaite à l'avance.

Code d’état HTTP : 412

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

ListJournalKinesisStreamsForLedger

Service : Amazon QLDB

Renvoie tous les flux de journaux Amazon QLDB pour un registre donné.

Cette action ne renvoie aucun flux de journal expiré. Pour plus d'informations, consultez la section Expiration des flux de terminaux dans le manuel Amazon QLDB Developer Guide.

Cette action renvoie un maximum d'`MaxResults`éléments. Il est paginé afin que vous puissiez récupérer tous les éléments en appelant `ListJournalKinesisStreamsForLedger` plusieurs fois.

Syntaxe de la demande

```
GET /ledgers/name/journal-kinesis-streams?max_results=MaxResults&next_token=NextToken
 HTTP/1.1
```

Paramètres de demande URI

La demande utilise les paramètres URI suivants.

name

Nom du registre.

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Modèle : `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatoire : oui

MaxResults

Le nombre maximum de résultats à renvoyer dans une seule `ListJournalKinesisStreamsForLedger` requête. (Le nombre réel de résultats renvoyés peut être inférieur.)

Plage valide : valeur minimum de 1. Valeur maximale fixée à 100.

NextToken

Un jeton de pagination, indiquant que vous souhaitez récupérer la page de résultats suivante. Si vous avez reçu une valeur pour `NextToken` dans la réponse d'un

`ListJournalKinesisStreamsForLedger` appel précédent, vous devez utiliser cette valeur comme entrée ici.

Contraintes de longueur : longueur minimale de 4. Longueur maximum de 1024.

Modèle : `^[A-Za-z-0-9+/=]+$`

Corps de la demande

La demande n'a pas de corps de requête.

Syntaxe de la réponse

```
HTTP/1.1 200
Content-type: application/json

{
   "NextToken": "string",
   "Streams": [
      {
         "Arn": "string",
         "CreationTime": number,
         "ErrorCause": "string",
         "ExclusiveEndTime": number,
         "InclusiveStartTime": number,
         "KinesisConfiguration": {
            "AggregationEnabled": boolean,
            "StreamArn": "string"
         },
         "LedgerName": "string",
         "RoleArn": "string",
         "Status": "string",
         "StreamId": "string",
         "StreamName": "string"
      }
   ]
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées au format JSON par le service.

NextToken

- S'il `NextToken` est vide, la dernière page de résultats a été traitée et il n'y a plus de résultats à récupérer.
- Si `NextToken` ce n'est pas vide, d'autres résultats sont disponibles. Pour récupérer la page de résultats suivante, utilisez la valeur de `NextToken` lors d'un `ListJournalKinesisStreamsForLedger` appel suivant.

Type : chaîne

Contraintes de longueur : longueur minimale de 4. Longueur maximum de 1024.

Modèle : `^[A-Za-z-0-9+/=]+$`

Streams

Les flux de journaux QLDB actuellement associés au registre donné.

Type : tableau d'objets JournalKinesisStreamDescription

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

InvalidParameterException

Un ou plusieurs paramètres de la demande ne sont pas valides.

Code d'état HTTP : 400

ResourceNotFoundException

La ressource spécifiée n'existe pas.

Code d'état HTTP : 404

ResourcePreconditionNotMetException

L'opération a échoué car une condition n'était pas satisfaite à l'avance.

Code d'état HTTP : 412

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

ListJournalS3Exports

Service : Amazon QLDB

Renvoie toutes les tâches d'exportation de journaux pour tous les livres associés au livre actuel Compte AWS et à la région.

Cette action renvoie un maximum d'`MaxResults`éléments et est paginée afin que vous puissiez récupérer tous les éléments en appelant `ListJournalS3Exports` plusieurs fois.

Cette action ne renvoie aucune tâche d'exportation expirée. Pour plus d'informations, consultez Exporter l'expiration des tâches dans le manuel Amazon QLDB Developer Guide.

Syntaxe de la demande

```
GET /journal-s3-exports?max_results=MaxResults&next_token=NextToken HTTP/1.1
```

Paramètres de demande URI

La demande utilise les paramètres URI suivants.

MaxResults

Le nombre maximum de résultats à renvoyer dans une seule `ListJournalS3Exports` requête. (Le nombre réel de résultats renvoyés peut être inférieur.)

Plage valide : valeur minimum de 1. Valeur maximale fixée à 100.

NextToken

Un jeton de pagination, indiquant que vous souhaitez récupérer la page de résultats suivante. Si vous avez reçu une valeur pour `NextToken` dans la réponse d'un `ListJournalS3Exports` appel précédent, vous devez utiliser cette valeur comme entrée ici.

Contraintes de longueur : longueur minimale de 4. Longueur maximum de 1024.

Modèle : `^[A-Za-z-0-9+/=]+$`

Corps de la demande

La demande n’a pas de corps de requête.

Syntaxe de la réponse

```
HTTP/1.1 200
Content-type: application/json

{
   "JournalS3Exports": [
      {
         "ExclusiveEndTime": number,
         "ExportCreationTime": number,
         "ExportId": "string",
         "InclusiveStartTime": number,
         "LedgerName": "string",
         "OutputFormat": "string",
         "RoleArn": "string",
         "S3ExportConfiguration": {
            "Bucket": "string",
            "EncryptionConfiguration": {
               "KmsKeyArn": "string",
               "ObjectEncryptionType": "string"
            },
            "Prefix": "string"
         },
         "Status": "string"
      }
   ],
   "NextToken": "string"
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées au format JSON par le service.

JournalS3Exports

Les tâches d'exportation du journal pour tous les livres associés au livre actuel Compte AWS et à la région.

Type : tableau d'objets JournalS3ExportDescription

NextToken

- S'il `NextToken` est vide, cela signifie que la dernière page de résultats a été traitée et qu'il n'y a plus de résultats à récupérer.
- Si `NextToken` ce n'est pas vide, cela signifie que d'autres résultats sont disponibles. Pour récupérer la page de résultats suivante, utilisez la valeur de `NextToken` lors d'un `ListJournalS3Exports` appel suivant.

Type : chaîne

Contraintes de longueur : longueur minimale de 4. Longueur maximum de 1024.

Modèle : `^[A-Za-z-0-9+/=]+$`

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

ListJournalS3ExportsForLedger

Service : Amazon QLDB

Renvoie toutes les tâches d'exportation de journaux pour un registre spécifié.

Cette action renvoie un maximum d'`MaxResults`éléments et est paginée afin que vous puissiez récupérer tous les éléments en appelant `ListJournalS3ExportsForLedger` plusieurs fois.

Cette action ne renvoie aucune tâche d'exportation expirée. Pour plus d'informations, consultez Exporter l'expiration des tâches dans le manuel Amazon QLDB Developer Guide.

Syntaxe de la demande

```
GET /ledgers/name/journal-s3-exports?max_results=MaxResults&next_token=NextToken
 HTTP/1.1
```

Paramètres de demande URI

La demande utilise les paramètres URI suivants.

MaxResults

Le nombre maximum de résultats à renvoyer dans une seule `ListJournalS3ExportsForLedger` requête. (Le nombre réel de résultats renvoyés peut être inférieur.)

Plage valide : valeur minimum de 1. Valeur maximale fixée à 100.

name

Nom du registre.

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Modèle : `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatoire : oui

NextToken

Un jeton de pagination, indiquant que vous souhaitez récupérer la page de résultats suivante. Si vous avez reçu une valeur pour `NextToken` dans la réponse d'un

`ListJournalS3ExportsForLedger` appel précédent, vous devez utiliser cette valeur comme entrée ici.

Contraintes de longueur : longueur minimale de 4. Longueur maximum de 1024.

Modèle : `^[A-Za-z-0-9+/=]+$`

Corps de la demande

La demande n'a pas de corps de requête.

Syntaxe de la réponse

```
HTTP/1.1 200
Content-type: application/json

{
   "JournalS3Exports": [
      {
         "ExclusiveEndTime": number,
         "ExportCreationTime": number,
         "ExportId": "string",
         "InclusiveStartTime": number,
         "LedgerName": "string",
         "OutputFormat": "string",
         "RoleArn": "string",
         "S3ExportConfiguration": {
            "Bucket": "string",
            "EncryptionConfiguration": {
               "KmsKeyArn": "string",
               "ObjectEncryptionType": "string"
            },
            "Prefix": "string"
         },
         "Status": "string"
      }
   ],
   "NextToken": "string"
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées au format JSON par le service.

JournalS3Exports

Les tâches d'exportation de journaux actuellement associées au registre spécifié.

Type : tableau d'objets JournalS3ExportDescription

NextToken

- S'il `NextToken` est vide, cela signifie que la dernière page de résultats a été traitée et qu'il n'y a plus de résultats à récupérer.
- Si `NextToken` ce n'est pas vide, cela signifie que d'autres résultats sont disponibles. Pour récupérer la page de résultats suivante, utilisez la valeur de `NextToken` lors d'un `ListJournalS3ExportsForLedger` appel suivant.

Type : chaîne

Contraintes de longueur : longueur minimale de 4. Longueur maximum de 1024.

Modèle : `^[A-Za-z-0-9+/=]+$`

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python

- AWS SDK pour Ruby V3

ListLedgers

Service : Amazon QLDB

Renvoie tous les registres associés au livre actuel Compte AWS et à la région.

Cette action renvoie un maximum d'`MaxResults`éléments et est paginée afin que vous puissiez récupérer tous les éléments en appelant `ListLedgers` plusieurs fois.

Syntaxe de la demande

```
GET /ledgers?max_results=MaxResults&next_token=NextToken HTTP/1.1
```

Paramètres de demande URI

La demande utilise les paramètres URI suivants.

MaxResults

Le nombre maximum de résultats à renvoyer dans une seule `ListLedgers` requête. (Le nombre réel de résultats renvoyés peut être inférieur.)

Plage valide : valeur minimum de 1. Valeur maximale fixée à 100.

NextToken

Un jeton de pagination, indiquant que vous souhaitez récupérer la page de résultats suivante. Si vous avez reçu une valeur pour `NextToken` dans la réponse d'un `ListLedgers` appel précédent, vous devez utiliser cette valeur comme entrée ici.

Contraintes de longueur : longueur minimale de 4. Longueur maximum de 1024.

Modèle : `^[A-Za-z-0-9+/=]+$`

Corps de la demande

La demande n’a pas de corps de requête.

Syntaxe de la réponse

```
HTTP/1.1 200
Content-type: application/json
```

```
{
   "Ledgers": [
      {
         "CreationDateTime": number,
         "Name": "string",
         "State": "string"
      }
   ],
   "NextToken": "string"
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées au format JSON par le service.

Ledgers

Les registres associés au courant Compte AWS et à la région.

Type : tableau d'objets LedgerSummary

NextToken

Un jeton de pagination, indiquant si d'autres résultats sont disponibles :

- S'il `NextToken` est vide, cela signifie que la dernière page de résultats a été traitée et qu'il n'y a plus de résultats à récupérer.
- Si `NextToken` ce n'est pas vide, cela signifie que d'autres résultats sont disponibles. Pour récupérer la page de résultats suivante, utilisez la valeur de `NextToken` lors d'un `ListLedgers` appel suivant.

Type : chaîne

Contraintes de longueur : longueur minimale de 4. Longueur maximum de 1024.

Modèle : `^[A-Za-z-0-9+/=]+$`

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

ListTagsForResource

Service : Amazon QLDB

Renvoie toutes les balises pour une ressource Amazon QLDB spécifiée.

Syntaxe de la demande

```
GET /tags/resourceArn HTTP/1.1
```

Paramètres de demande URI

La demande utilise les paramètres URI suivants.

resourceArn

Nom de ressource Amazon (ARN) pour lequel répertorier les balises. Par exemple :

`arn:aws:qldb:us-east-1:123456789012:ledger/exampleLedger`

Contraintes de longueur : longueur minimale de 20. Longueur maximale de 1600.

Obligatoire : oui

Corps de la demande

La demande n'a pas de corps de requête.

Syntaxe de la réponse

```
HTTP/1.1 200
Content-type: application/json

{
   "Tags": {
      "string" : "string"
   }
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées au format JSON par le service.

Tags

Les balises actuellement associées à la ressource Amazon QLDB spécifiée.

Type : mappage chaîne/chaîne

Entrées cartographiques : nombre minimum de 0 éléments. Nombre maximum de 200 éléments.

Contraintes de longueur de clé : longueur minimale de 1. Longueur maximale de 128.

Contraintes de longueur de valeur : longueur minimale de 0. Longueur maximale de 256.

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

InvalidParameterException

Un ou plusieurs paramètres de la demande ne sont pas valides.

Code d’état HTTP : 400

ResourceNotFoundException

La ressource spécifiée n’existe pas.

Code d’état HTTP : 404

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2

- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

StreamJournalToKinesis

Service : Amazon QLDB

Crée un flux de journal pour un registre Amazon QLDB donné. Le flux capture chaque révision de document validée dans le journal du registre et transmet les données à une ressource Amazon Kinesis Data Streams spécifiée.

Syntaxe de la demande

```
POST /ledgers/name/journal-kinesis-streams HTTP/1.1
Content-type: application/json

{
   "ExclusiveEndTime": number,
   "InclusiveStartTime": number,
   "KinesisConfiguration": {
      "AggregationEnabled": boolean,
      "StreamArn": "string"
   },
   "RoleArn": "string",
   "StreamName": "string",
   "Tags": {
      "string" : "string"
   }
}
```

Paramètres de demande URI

La demande utilise les paramètres URI suivants.

name

Nom du registre.

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Modèle : `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatoire : oui

Corps de la demande

Cette demande accepte les données suivantes au format JSON.

ExclusiveEndTime

Date et heure exclusives spécifiant la date à laquelle le flux se termine. Si vous ne définissez pas ce paramètre, le flux s'exécute indéfiniment jusqu'à ce que vous l'annuliez.

Le paramètre `ExclusiveEndTime` doit être au format date et heure `ISO 8601` et en temps universel coordonné (UTC). Par exemple : `2019-06-13T21:36:34Z`.

Type : Timestamp

Obligatoire : non

InclusiveStartTime

Date et heure de début incluses à partir de laquelle commence la diffusion des données de journal. Ce paramètre doit être au format date et heure `ISO 8601` et au format temps universel coordonné (UTC). Par exemple : `2019-06-13T21:36:34Z`.

Le paramètre `InclusiveStartTime` ne peut pas être dans le futur et doit être avant `ExclusiveEndTime`.

Si vous fournissez un paramètre `InclusiveStartTime` qui se trouve avant le registre `CreationDateTime`, QLDB l'affecte par défaut au paramètre `CreationDateTime` du registre.

Type : Timestamp

Obligatoire : oui

KinesisConfiguration

Paramètres de configuration de la destination Kinesis Data Streams pour votre demande de flux.

Type : objet KinesisConfiguration

Obligatoire : oui

RoleArn

Amazon Resource Name (ARN) du rôle IAM qui accorde des autorisations QLDB à un flux de journal pour écrire des enregistrements de données dans une ressource Kinesis Data Streams.

Pour transmettre un rôle à QLDB lorsque vous demandez un flux de journal, vous devez disposer des autorisations nécessaires pour effectuer l'action `iam:PassRole` sur la ressource de rôle IAM. Cela est obligatoire pour toutes les demandes de flux de journal.

Type : chaîne

Contraintes de longueur : longueur minimale de 20. Longueur maximale de 1600.

Obligatoire : oui

StreamName

Nom que vous souhaitez affecter au flux de journal QLDB. Les noms définis par l'utilisateur peuvent aider à identifier et à indiquer le but d'un flux.

Votre nom de flux doit être unique parmi les autres flux actifs pour un registre donné. Les contraintes relatives aux noms de flux sont les mêmes que celles relatives aux noms de registres, telles que définies dans Quotas dans Amazon QLDB dans le Manuel du développeur Amazon QLDB.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Modèle : `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatoire : oui

Tags

Les paires clé-valeur à ajouter sous forme de balises au flux que vous souhaitez créer. Les clés de balises sont sensibles à la casse. Les valeurs des balises distinguent les majuscules et minuscules et peuvent être nulles.

Type : mappage chaîne/chaîne

Entrées cartographiques : nombre minimum de 0 éléments. Nombre maximum de 200 éléments.

Contraintes de longueur de clé : longueur minimale de 1. Longueur maximale de 128.

Contraintes de longueur de valeur : longueur minimale de 0. Longueur maximale de 256.

Obligatoire : non

Syntaxe de la réponse

```
HTTP/1.1 200
```

```
Content-type: application/json

{
   "StreamId": "string"
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées au format JSON par le service.

StreamId

L'UUID (représenté dans du texte codé en Base62) que QLDB attribue à chaque flux de journal QLDB.

Type : chaîne

Contraintes de longueur : longueur fixe de 22.

Modèle : `^[A-Za-z-0-9]+$`

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

InvalidParameterException

Un ou plusieurs paramètres de la demande ne sont pas valides.

Code d'état HTTP : 400

ResourceNotFoundException

La ressource spécifiée n'existe pas.

Code d'état HTTP : 404

ResourcePreconditionNotMetException

L'opération a échoué car une condition n'était pas satisfaite à l'avance.

Code d'état HTTP : 412

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

TagResource

Service : Amazon QLDB

Ajoute une ou plusieurs balises à une ressource Amazon QLDB spécifiée.

Une ressource peut comporter jusqu'à 50 balises. Si vous essayez de créer plus de 50 balises pour une ressource, votre demande échoue et renvoie une erreur.

Syntaxe de la demande

```
POST /tags/resourceArn HTTP/1.1
Content-type: application/json

{
   "Tags": {
      "string" : "string"
   }
}
```

Paramètres de demande URI

La demande utilise les paramètres URI suivants.

resourceArn

Le nom de ressource Amazon (ARN) auquel vous souhaitez ajouter les balises. Par exemple :

`arn:aws:qldb:us-east-1:123456789012:ledger/exampleLedger`

Contraintes de longueur : longueur minimale de 20. Longueur maximale de 1600.

Obligatoire : oui

Corps de la demande

Cette demande accepte les données suivantes au format JSON.

Tags

Les paires clé-valeur à ajouter sous forme de balises à la ressource QLDB spécifiée. Les clés de balises sont sensibles à la casse. Si vous spécifiez une clé qui existe déjà pour la ressource,

votre demande échoue et renvoie une erreur. Les valeurs des balises distinguent les majuscules et minuscules et peuvent être nulles.

Type : mappage chaîne/chaîne

Entrées cartographiques : nombre minimum de 0 éléments. Nombre maximum de 200 éléments.

Contraintes de longueur de clé : longueur minimale de 1. Longueur maximale de 128.

Contraintes de longueur de valeur : longueur minimale de 0. Longueur maximale de 256.

Obligatoire : oui

Syntaxe de la réponse

```
HTTP/1.1 200
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200 avec un corps HTTP vide.

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

InvalidParameterException

Un ou plusieurs paramètres de la demande ne sont pas valides.

Code d'état HTTP : 400

ResourceNotFoundException

La ressource spécifiée n'existe pas.

Code d'état HTTP : 404

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

UntagResource

Service : Amazon QLDB

Supprime une ou plusieurs balises d'une ressource Amazon QLDB spécifiée. Vous pouvez spécifier jusqu'à 50 clés de balise à supprimer.

Syntaxe de la demande

```
DELETE /tags/resourceArn?tagKeys=TagKeys HTTP/1.1
```

Paramètres de demande URI

La demande utilise les paramètres URI suivants.

resourceArn

Le nom de ressource Amazon (ARN) à partir duquel les balises doivent être supprimées. Par exemple :

`arn:aws:qldb:us-east-1:123456789012:ledger/exampleLedger`

Contraintes de longueur : longueur minimale de 20. Longueur maximale de 1600.

Obligatoire : oui

TagKeys

La liste des clés de balise à supprimer.

Membres du tableau : nombre minimum de 0 élément. Nombre maximum de 200 éléments.

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 128.

Obligatoire : oui

Corps de la demande

La demande n’a pas de corps de requête.

Syntaxe de la réponse

```
HTTP/1.1 200
```

Eléments de réponse

Si l’action aboutit, le service renvoie une réponse HTTP 200 avec un corps HTTP vide.

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

InvalidParameterException

Un ou plusieurs paramètres de la demande ne sont pas valides.

Code d’état HTTP : 400

ResourceNotFoundException

La ressource spécifiée n’existe pas.

Code d’état HTTP : 404

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

UpdateLedger

Service : Amazon QLDB

Met à jour les propriétés d'un registre.

Syntaxe de la demande

```
PATCH /ledgers/name HTTP/1.1
Content-type: application/json

{
   "DeletionProtection": boolean,
   "KmsKey": "string"
}
```

Paramètres de demande URI

La demande utilise les paramètres URI suivants.

name

Nom du registre.

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Modèle : `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatoire : oui

Corps de la demande

Cette demande accepte les données suivantes au format JSON.

DeletionProtection

Spécifie si le registre est protégé contre la suppression par un utilisateur. À défaut de définition lors de la création du registre, cette fonctionnalité est activée (`true`) par défaut.

Si la protection contre la suppression est activée, vous devez commencer par la désactiver avant de pouvoir supprimer le registre. Vous pouvez la désactiver en appelant l'opération `UpdateLedger` pour définir ce paramètre sur `false`.

Type : booléen

Obligatoire : non

KmsKey

La clé in AWS Key Management Service (AWS KMS) à utiliser pour le chiffrement des données au repos dans le registre. Pour plus d'informations, veuillez consulter la rubrique Chiffrement au repos dans le Guide du développeur Amazon QLDB.

Utilisez l'une des options suivantes pour spécifier ce paramètre :

- `AWS_OWNED_KMS_KEY`: utilisez une AWS KMS clé détenue et gérée par AWS vous.
- Non défini : n'apportez aucune modification à la clé KMS du registre.
- Une clé KMS symétrique gérée par le client valide : utilisez la clé KMS de chiffrement symétrique spécifiée dans votre compte que vous créez, possédez et gérez.

  Amazon QLDB ne prend pas en charge les clés asymétriques. Pour plus d'informations, consultez la section Utilisation de clés symétriques et asymétriques dans le manuel du AWS Key Management Service développeur.

Pour spécifier une clé KMS gérée par le client, utilisez son ID de clé, son Amazon Resource Name (ARN), son nom d'alias ou son ARN d'alias. Lorsque vous utilisez un nom d'alias, préfixez-le avec `"alias/"`. Pour spécifier une clé dans un autre Compte AWS, vous devez utiliser l'ARN de la clé ou l'alias ARN.

Par exemple :

- ID de clé : `1234abcd-12ab-34cd-56ef-1234567890ab`
- ARN de clé : `arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`
- Nom d'alias : `alias/ExampleAlias`
- ARN d'alias : `arn:aws:kms:us-east-2:111122223333:alias/ExampleAlias`

Pour plus d'informations, consultez la section Identifiants clés (KeyId) dans le guide du AWS Key Management Service développeur.

Type : chaîne

Contraintes de longueur : longueur maximale de 1600.

Obligatoire : non

Syntaxe de la réponse

```
HTTP/1.1 200
Content-type: application/json

{
   "Arn": "string",
   "CreationDateTime": number,
   "DeletionProtection": boolean,
   "EncryptionDescription": {
      "EncryptionStatus": "string",
      "InaccessibleKmsKeyDateTime": number,
      "KmsKeyArn": "string"
   },
   "Name": "string",
   "State": "string"
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées au format JSON par le service.

Arn

Le nom de ressource Amazon (ARN) du registre.

Type : chaîne

Contraintes de longueur : longueur minimale de 20. Longueur maximale de 1600.

CreationDateTime

Date et heure, au format epoch time, auxquelles le registre a été créé. (Le format Epoch est le nombre de secondes écoulées depuis 00h00 le 1er janvier 1970 UTC.)

Type : Timestamp

DeletionProtection

Spécifie si le registre est protégé contre la suppression par un utilisateur. À défaut de définition lors de la création du registre, cette fonctionnalité est activée (`true`) par défaut.

Si la protection contre la suppression est activée, vous devez commencer par la désactiver avant de pouvoir supprimer le registre. Vous pouvez la désactiver en appelant l'opération `UpdateLedger` pour définir ce paramètre sur `false`.

Type : booléen

EncryptionDescription

Informations sur le chiffrement des données au repos dans le registre. Cela inclut l'état actuel, la AWS KMS clé et le moment où la clé est devenue inaccessible (en cas d'erreur).

Type : objet LedgerEncryptionDescription

Name

Nom du registre.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Modèle : `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

State

État actuel du registre.

Type : chaîne

Valeurs valides : `CREATING | ACTIVE | DELETING | DELETED`

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

InvalidParameterException

Un ou plusieurs paramètres de la demande ne sont pas valides.

Code d’état HTTP : 400

ResourceNotFoundException

La ressource spécifiée n’existe pas.

Code d’état HTTP : 404

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

UpdateLedgerPermissionsMode

Service : Amazon QLDB

Met à jour le mode d'autorisations d'un registre.

> **Important**
>
> Avant de passer en mode STANDARD autorisations, vous devez d'abord créer toutes les politiques IAM et les balises de table requises afin de ne pas perturber la vie de vos utilisateurs. Pour en savoir plus, consultez la section Migration vers le mode d'autorisations standard dans le manuel Amazon QLDB Developer Guide.

Syntaxe de la demande

```
PATCH /ledgers/name/permissions-mode HTTP/1.1
Content-type: application/json

{
   "PermissionsMode": "string"
}
```

Paramètres de demande URI

La demande utilise les paramètres URI suivants.

name

Nom du registre.

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Modèle : `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatoire : oui

Corps de la demande

Cette demande accepte les données suivantes au format JSON.

PermissionsMode

Mode d'autorisations à attribuer au registre. Ce paramètre peut avoir une des valeurs suivantes :

- `ALLOW_ALL` : un mode d'autorisations hérité qui permet le contrôle d'accès avec une granularité au niveau de l'API pour les registres.

  Ce mode permet aux utilisateurs qui possèdent l'autorisation d'API `SendCommand` pour ce registre d'exécuter toutes les commandes PartiQL (par conséquent, `ALLOW_ALL`) sur toutes les tables du registre spécifié. Ce mode ignore les politiques d'autorisations IAM au niveau de la table ou de la commande que vous créez pour le registre.
- `STANDARD` : (Recommandé) un mode d'autorisations qui permet le contrôle d'accès avec une granularité plus fine pour les registres, les tables et les commandes PartiQL.

  Par défaut, ce mode refuse toutes les demandes utilisateur d'exécuter des commandes PartiQL sur les tables de ce registre. Pour autoriser l'exécution des commandes PartiQL, vous devez créer des politiques d'autorisations IAM pour des ressources de table spécifiques et des actions PartiQL, en plus de l'autorisation d'API `SendCommand` pour le registre. Pour plus d'informations, veuillez consulter la rubrique Premiers pas avec le mode d'autorisations standard dans le Guide du développeur Amazon QLDB.

> **Note**
>
> Nous vous recommandons vivement d'utiliser le mode d'autorisations `STANDARD` pour optimiser la sécurité des données de votre registre.

Type : chaîne

Valeurs valides : `ALLOW_ALL | STANDARD`

Obligatoire : oui

Syntaxe de la réponse

```
HTTP/1.1 200
Content-type: application/json

{
   "Arn": "string",
```

```
    "Name": "string",
    "PermissionsMode": "string"
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées au format JSON par le service.

Arn

Le nom de ressource Amazon (ARN) du registre.

Type : chaîne

Contraintes de longueur : longueur minimale de 20. Longueur maximale de 1600.

Name

Nom du registre.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Modèle : `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

PermissionsMode

Le mode d'autorisations actuel du registre.

Type : chaîne

Valeurs valides : `ALLOW_ALL | STANDARD`

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

InvalidParameterException

Un ou plusieurs paramètres de la demande ne sont pas valides.

Code d’état HTTP : 400

ResourceNotFoundException

La ressource spécifiée n’existe pas.

Code d’état HTTP : 404

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

Session Amazon QLDB

Les actions suivantes sont prises en charge par la session Amazon QLDB

- SendCommand

SendCommand

Service : Amazon QLDB Session

Envoie une commande à un registre Amazon QLDB.

> **Note**
>
> Au lieu d'interagir directement avec cette API, nous vous recommandons d'utiliser le pilote QLDB ou le shell QLDB pour exécuter des transactions de données sur un registre.
>
> - Si vous travaillez avec un AWS SDK, utilisez le pilote QLDB. Le pilote fournit une couche d'abstraction de haut niveau au-dessus de cette API de données de session QLDB et gère `SendCommand` l'opération pour vous. Pour obtenir des informations et une liste des langages de programmation pris en charge, consultez Getting started with the driver dans le manuel Amazon QLDB Developer Guide.
> - Si vous travaillez avec le AWS Command Line Interface (AWS CLI), utilisez le shell QLDB. Le shell est une interface de ligne de commande qui utilise le pilote QLDB pour interagir avec un registre. Pour plus d'informations, consultez la section Accès à Amazon QLDB à l'aide du shell QLDB.

Syntaxe de la requête

```
{
   "AbortTransaction": {
   },
   "CommitTransaction": {
      "CommitDigest": blob,
      "TransactionId": "string"
   },
   "EndSession": {
   },
   "ExecuteStatement": {
      "Parameters": [
         {
            "IonBinary": blob,
            "IonText": "string"
         }
      ],
      "Statement": "string",
```

```
      "TransactionId": "string"
   },
   "FetchPage": {
      "NextPageToken": "string",
      "TransactionId": "string"
   },
   "SessionToken": "string",
   "StartSession": {
      "LedgerName": "string"
   },
   "StartTransaction": {
   }
}
```

Paramètres de demande

Pour plus d'informations sur les paramètres courants pour toutes les actions, consultez Paramètres courants.

Cette demande accepte les données suivantes au format JSON.

AbortTransaction

Commande pour annuler la transaction en cours.

Type : objet AbortTransactionRequest

Obligatoire : non

CommitTransaction

Commande permettant de valider la transaction spécifiée.

Type : objet CommitTransactionRequest

Obligatoire : non

EndSession

Commande pour mettre fin à la session en cours.

Type : objet EndSessionRequest

Obligatoire : non

ExecuteStatement

Commande permettant d'exécuter une instruction dans la transaction spécifiée.

Type : objet ExecuteStatementRequest

Obligatoire : non

FetchPage

Commande pour récupérer une page.

Type : objet FetchPageRequest

Obligatoire : non

SessionToken

Spécifie le jeton de session pour la commande en cours. Un jeton de session est constant pendant toute la durée de vie de la session.

Pour obtenir un jeton de session, exécutez la `StartSession` commande. Cela `SessionToken` est obligatoire pour chaque commande suivante émise au cours de la session en cours.

Type : chaîne

Contraintes de longueur : longueur minimale de 4. Longueur maximum de 1024.

Modèle : `^[A-Za-z-0-9+/=]+$`

Obligatoire : non

StartSession

Commande pour démarrer une nouvelle session. Un jeton de session est obtenu dans le cadre de la réponse.

Type : objet StartSessionRequest

Obligatoire : non

StartTransaction

Commande pour démarrer une nouvelle transaction.

Type : objet StartTransactionRequest

Obligatoire : non

Syntaxe de la réponse

```
{
   "AbortTransaction": {
      "TimingInformation": {
         "ProcessingTimeMilliseconds": number
      }
   },
   "CommitTransaction": {
      "CommitDigest": blob,
      "ConsumedIOs": {
         "ReadIOs": number,
         "WriteIOs": number
      },
      "TimingInformation": {
         "ProcessingTimeMilliseconds": number
      },
      "TransactionId": "string"
   },
   "EndSession": {
      "TimingInformation": {
         "ProcessingTimeMilliseconds": number
      }
   },
   "ExecuteStatement": {
      "ConsumedIOs": {
         "ReadIOs": number,
         "WriteIOs": number
      },
      "FirstPage": {
         "NextPageToken": "string",
         "Values": [
            {
               "IonBinary": blob,
               "IonText": "string"
            }
         ]
      },
      "TimingInformation": {
         "ProcessingTimeMilliseconds": number
      }
   },
   "FetchPage": {
      "ConsumedIOs": {
```

```
            "ReadIOs": number,
            "WriteIOs": number
         },
         "Page": {
            "NextPageToken": "string",
            "Values": [
               {
                  "IonBinary": blob,
                  "IonText": "string"
               }
            ]
         },
         "TimingInformation": {
            "ProcessingTimeMilliseconds": number
         }
      },
      "StartSession": {
         "SessionToken": "string",
         "TimingInformation": {
            "ProcessingTimeMilliseconds": number
         }
      },
      "StartTransaction": {
         "TimingInformation": {
            "ProcessingTimeMilliseconds": number
         },
         "TransactionId": "string"
      }
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées au format JSON par le service.

AbortTransaction

Contient les détails de la transaction abandonnée.

Type : objet AbortTransactionResult

CommitTransaction

Contient les détails de la transaction validée.

Type : objet CommitTransactionResult

EndSession

Contient les détails de la session terminée.

Type : objet EndSessionResult

ExecuteStatement

Contient les détails de l'instruction exécutée.

Type : objet ExecuteStatementResult

FetchPage

Contient les détails de la page récupérée.

Type : objet FetchPageResult

StartSession

Contient les détails de la session démarrée, y compris un jeton de session. Cela `SessionToken` est obligatoire pour chaque commande suivante émise au cours de la session en cours.

Type : objet StartSessionResult

StartTransaction

Contient les détails de la transaction démarrée.

Type : objet StartTransactionResult

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

BadRequestException

Renvoyé si la demande est mal formulée ou contient une erreur telle qu'une valeur de paramètre non valide ou un paramètre obligatoire manquant.

Code d'état HTTP : 400

CapacityExceededException

Renvoyé lorsque la demande dépasse la capacité de traitement du registre.

Code d'état HTTP : 400

InvalidSessionException

Renvoyé si la session n'existe plus parce qu'elle a expiré ou a expiré.

Code d'état HTTP : 400

LimitExceededException

Renvoie si une limite de ressources, telle que le nombre de sessions actives, est dépassée.

Code d'état HTTP : 400

OccConflictException

Renvoyé lorsqu'une transaction ne peut pas être écrite dans le journal en raison d'un échec lors de la phase de vérification du contrôle de simultanéité optimiste (OCC).

Code d'état HTTP : 400

RateExceededException

Renvoyé lorsque le taux de demandes dépasse le débit autorisé.

Code d'état HTTP : 400

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3

- AWS SDK pour Python
- AWS SDK pour Ruby V3

Types de données

Les types de données suivants sont pris en charge par Amazon QLDB :

- JournalKinesisStreamDescription
- JournalS3ExportDescription
- KinesisConfiguration
- LedgerEncryptionDescription
- LedgerSummary
- S3EncryptionConfiguration
- S3ExportConfiguration
- ValueHolder

Les types de données suivants sont pris en charge par Amazon QLDB Session :

- AbortTransactionRequest
- AbortTransactionResult
- CommitTransactionRequest
- CommitTransactionResult
- EndSessionRequest
- EndSessionResult
- ExecuteStatementRequest
- ExecuteStatementResult
- FetchPageRequest
- FetchPageResult
- IOUsage
- Page
- StartSessionRequest
- StartSessionResult

- StartTransactionRequest
- StartTransactionResult
- TimingInformation
- ValueHolder

Amazon QLDB

The following data types are supported by Amazon QLDB:

- JournalKinesisStreamDescription
- JournalS3ExportDescription
- KinesisConfiguration
- LedgerEncryptionDescription
- LedgerSummary
- S3EncryptionConfiguration
- S3ExportConfiguration
- ValueHolder

JournalKinesisStreamDescription

Service : Amazon QLDB

Informations sur un flux de journal Amazon QLDB, notamment le nom de la ressource Amazon (ARN), le nom du flux, l'heure de création, le statut actuel et les paramètres de la demande de création de flux d'origine.

Table des matières

KinesisConfiguration

Les paramètres de configuration de la destination Amazon Kinesis Data Streams pour un flux de journal QLDB.

Type : objet KinesisConfiguration

Obligatoire : oui

LedgerName

Nom du registre.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Modèle : `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatoire : oui

RoleArn

Amazon Resource Name (ARN) du rôle IAM qui accorde des autorisations QLDB à un flux de journal pour écrire des enregistrements de données dans une ressource Kinesis Data Streams.

Type : chaîne

Contraintes de longueur : longueur minimale de 20. Longueur maximale de 1600.

Obligatoire : oui

Status

État actuel du flux de journal QLDB.

Type : chaîne

Valeurs valides : `ACTIVE | COMPLETED | CANCELED | FAILED | IMPAIRED`

Obligatoire : oui

StreamId

L'UUID (représenté dans le texte codé en Base62) du flux de journal QLDB.

Type : chaîne

Contraintes de longueur : longueur fixe de 22.

Modèle : `^[A-Za-z-0-9]+$`

Obligatoire : oui

StreamName

Nom défini par l'utilisateur du flux de journal QLDB.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Modèle : `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatoire : oui

Arn

L'Amazon Resource Name (ARN) du flux de journal QLDB.

Type : chaîne

Contraintes de longueur : longueur minimale de 20. Longueur maximale de 1600.

Obligatoire : non

CreationTime

Date et heure, au format Epoch Time, auxquelles le flux de journal QLDB a été créé. (Le format Epoch est le nombre de secondes écoulées depuis 00h00 le 1er janvier 1970 UTC.)

Type : Timestamp

Obligatoire : non

ErrorCause

Le message d'erreur qui décrit la raison pour laquelle le statut d'un flux est `IMPAIRED` ou`FAILED`. Cela ne s'applique pas aux flux qui ont d'autres valeurs de statut.

Type : chaîne

Valeurs valides : `KINESIS_STREAM_NOT_FOUND | IAM_PERMISSION_REVOKED`

Obligatoire : non

ExclusiveEndTime

Date et heure exclusives spécifiant la date à laquelle le flux se termine. Si ce paramètre n'est pas défini, le flux s'exécute indéfiniment jusqu'à ce que vous l'annuliez.

Type : Timestamp

Obligatoire : non

InclusiveStartTime

Date et heure de début incluses à partir de laquelle commence la diffusion des données de journal.

Type : Timestamp

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

JournalS3ExportDescription

Service : Amazon QLDB

Informations sur une tâche d'exportation de journal, notamment le nom du registre, l'ID d'exportation, l'heure de création, le statut actuel et les paramètres de la demande de création d'exportation d'origine.

Table des matières

ExclusiveEndTime

Date et heure de fin exclusives pour la gamme de contenus du journal spécifiée dans la demande d'exportation initiale.

Type : Timestamp

Obligatoire : oui

ExportCreationTime

Date et heure, au format epoch time, auxquelles la tâche d'exportation a été créée. (Le format Epoch est le nombre de secondes écoulées depuis 00h00 le 1er janvier 1970 UTC.)

Type : Timestamp

Obligatoire : oui

ExportId

L'UUID (représenté dans le texte codé en Base62) de la tâche d'exportation du journal.

Type : chaîne

Contraintes de longueur : longueur fixe de 22.

Modèle : `^[A-Za-z-0-9]+$`

Obligatoire : oui

InclusiveStartTime

Date et heure de début incluses pour la gamme de contenus du journal spécifiée dans la demande d'exportation initiale.

Type : Timestamp

Obligatoire : oui

LedgerName

Nom du registre.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Modèle : `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatoire : oui

RoleArn

Le nom de ressource Amazon (ARN) du rôle IAM qui accorde les autorisations QLDB pour une tâche d'exportation de journal permet d'effectuer les opérations suivantes :

- Écrivez des objets dans votre compartiment Amazon Simple Storage Service (Amazon S3).
- (Facultatif) Utilisez votre clé gérée par le client dans AWS Key Management Service (AWS KMS) pour le chiffrement côté serveur de vos données exportées.

Type : chaîne

Contraintes de longueur : longueur minimale de 20. Longueur maximale de 1600.

Obligatoire : oui

S3ExportConfiguration

Emplacement du compartiment Amazon Simple Storage Service (Amazon S3) dans lequel une tâche d'exportation de journal écrit le contenu du journal.

Type : objet S3ExportConfiguration

Obligatoire : oui

Status

État actuel de la tâche d'exportation du journal.

Type : chaîne

Valeurs valides : `IN_PROGRESS | COMPLETED | CANCELLED`

Obligatoire : oui

OutputFormat

Format de sortie des données de journal exportées.

Type : chaîne

Valeurs valides : `ION_BINARY | ION_TEXT | JSON`

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

KinesisConfiguration

Service : Amazon QLDB

Paramètres de configuration de la destination Amazon Kinesis Data Streams pour un flux de journal Amazon QLDB.

Table des matières

StreamArn

L'Amazon Resource Name (ARN) de la ressource Kinesis Data Streams.

Type : chaîne

Contraintes de longueur : longueur minimale de 20. Longueur maximale de 1600.

Obligatoire : oui

AggregationEnabled

Permet à QLDB de publier plusieurs enregistrements de données dans un seul enregistrement Kinesis Data Streams, augmentant ainsi le nombre d'enregistrements envoyés par appel d'API.

Par défaut : `True`

> **Important**
>
> Le groupement d'enregistrements a des implications importantes pour le traitement des enregistrements et nécessite un dégroupement dans votre consommateur de flux. Pour en savoir plus, consultez les sections KPL Concepts clés et Dégroupement côté consommateur dans le Manuel du développeur Amazon Kinesis Data Streams.

Type : booléen

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

LedgerEncryptionDescription

Service : Amazon QLDB

Informations sur le chiffrement des données au repos dans un registre Amazon QLDB. Cela inclut l'état actuel, la clé entrée AWS Key Management Service (AWS KMS) et le moment où la clé est devenue inaccessible (en cas d'erreur).

Pour plus d'informations, veuillez consulter la rubrique Chiffrement au repos dans le Guide du développeur Amazon QLDB.

Table des matières

EncryptionStatus

État actuel du chiffrement au repos pour le registre. Il peut s’agir de l’une des valeurs suivantes :

- `ENABLED`: le chiffrement est entièrement activé à l'aide de la clé spécifiée.
- `UPDATING`: Le registre traite activement le changement de clé spécifié.

  Les principales modifications apportées à QLDB sont asynchrones. Le registre est entièrement accessible sans aucun impact sur les performances pendant le traitement de la modification clé. Le temps nécessaire pour mettre à jour une clé varie en fonction de la taille du registre.
- `KMS_KEY_INACCESSIBLE`: La clé KMS gérée par le client spécifiée n'est pas accessible et le registre est altéré. Soit la clé a été désactivée ou supprimée, soit les autorisations associées à la clé ont été révoquées. Lorsqu'un registre est endommagé, il n'est pas accessible et n'accepte aucune demande de lecture ou d'écriture.

  Un registre altéré revient automatiquement à un état actif une fois que vous avez rétabli les autorisations sur la clé ou que vous avez réactivé la clé qui a été désactivée. Cependant, la suppression d'une clé KMS gérée par le client est irréversible. Une fois qu'une clé est supprimée, vous ne pouvez plus accéder aux registres protégés par cette clé, et les données deviennent définitivement irrécupérables.

Type : chaîne

Valeurs valides : `ENABLED | UPDATING | KMS_KEY_INACCESSIBLE`

Obligatoire : oui

KmsKeyArn

Le nom de ressource Amazon (ARN) de la clé KMS gérée par le client que le registre utilise pour le chiffrement au repos. Si ce paramètre n'est pas défini, le registre utilise une clé KMS AWS détenue pour le chiffrement. Il s'affichera `AWS_OWNED_KMS_KEY` lors de la mise à jour de la configuration de chiffrement du registre avec la clé KMS AWS détenue.

Type : chaîne

Contraintes de longueur : longueur minimale de 20. Longueur maximale de 1600.

Obligatoire : oui

InaccessibleKmsKeyDateTime

Date et heure, au format epoch time, auxquelles la AWS KMS clé est devenue inaccessible pour la première fois, en cas d'erreur. (Le format d'heure Epoch est le nombre de secondes qui se sont écoulées depuis 00h00 le 1er janvier 1970 UTC.)

Ce paramètre n'est pas défini si la AWS KMS clé est accessible.

Type : Timestamp

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

LedgerSummary

Service : Amazon QLDB

Informations sur un registre, notamment son nom, son état et sa date de création.

Table des matières

CreationDateTime

Date et heure, au format epoch time, auxquelles le registre a été créé. (Le format Epoch est le nombre de secondes écoulées depuis 00h00 le 1er janvier 1970 UTC.)

Type : Timestamp

Obligatoire : non

Name

Nom du registre.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Modèle : `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatoire : non

State

État actuel du registre.

Type : chaîne

Valeurs valides : `CREATING | ACTIVE | DELETING | DELETED`

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++

- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

S3EncryptionConfiguration

Service : Amazon QLDB

Les paramètres de chiffrement utilisés par une tâche d'exportation de journal pour écrire des données dans un compartiment Amazon Simple Storage Service (Amazon S3).

Table des matières

ObjectEncryptionType

Type de chiffrement des objets Amazon S3.

Pour en savoir plus sur les options de chiffrement côté serveur dans Amazon S3, consultez la section Protection des données à l'aide du chiffrement côté serveur dans le manuel du développeur Amazon S3.

Type : chaîne

Valeurs valides : `SSE_KMS | SSE_S3 | NO_ENCRYPTION`

Obligatoire : oui

KmsKeyArn

Le nom de ressource Amazon (ARN) d'une clé de chiffrement symétrique dans AWS Key Management Service (AWS KMS). Amazon S3 ne prend pas en charge les clés KMS asymétriques.

Vous devez fournir un `KmsKeyArn` si vous le spécifiez `SSE_KMS` comme`ObjectEncryptionType`.

`KmsKeyArn`n'est pas obligatoire si vous spécifiez `SSE_S3` comme`ObjectEncryptionType`.

Type : chaîne

Contraintes de longueur : longueur minimale de 20. Longueur maximale de 1600.

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

S3ExportConfiguration

Service : Amazon QLDB

Emplacement du compartiment Amazon Simple Storage Service (Amazon S3) dans lequel une tâche d'exportation de journal écrit le contenu du journal.

Table des matières

Bucket

Nom du compartiment Amazon S3 dans lequel une tâche d'exportation de journal écrit le contenu du journal.

Le nom du compartiment doit être conforme aux conventions de dénomination des compartiments Amazon S3. Pour plus d'informations, consultez la section Restrictions et limitations relatives aux compartiments dans le guide du développeur Amazon S3.

Type : chaîne

Contraintes de longueur : Longueur minimum de 3. Longueur maximale de 255.

Modèle : `^[A-Za-z-0-9-_.]+$`

Obligatoire : oui

EncryptionConfiguration

Les paramètres de chiffrement utilisés par une tâche d'exportation de journal pour écrire des données dans un compartiment Amazon S3.

Type : objet S3EncryptionConfiguration

Obligatoire : oui

Prefix

Préfixe du compartiment Amazon S3 dans lequel une tâche d'exportation de journal écrit le contenu du journal.

Le préfixe doit être conforme aux règles et restrictions de dénomination des clés Amazon S3. Pour plus d'informations, consultez la section Clé d'objet et métadonnées dans le manuel Amazon S3 Developer Guide.

Voici des exemples de `Prefix` valeurs valides :

- `JournalExports-ForMyLedger/Testing/`
- `JournalExports`
- `My:Tests/`

Type : chaîne

Contraintes de longueur : longueur minimum de 0. Longueur maximale de 128.

Obligatoire : oui

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

ValueHolder

Service : Amazon QLDB

Structure qui peut contenir une valeur dans plusieurs formats de codage.

Table des matières

IonText

Une valeur en texte brut Amazon Ion contenue dans une `ValueHolder` structure.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 1048576.

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

Session Amazon QLDB

Les types de données suivants sont pris en charge par Amazon QLDB :

- AbortTransactionRequest
- AbortTransactionResult
- CommitTransactionRequest
- CommitTransactionResult
- EndSessionRequest
- EndSessionResult
- ExecuteStatementRequest
- ExecuteStatementResult

- FetchPageRequest
- FetchPageResult
- IOUsage
- Page
- StartSessionRequest
- StartSessionResult
- StartTransactionRequest
- StartTransactionResult
- TimingInformation
- ValueHolder

AbortTransactionRequest

Service : Amazon QLDB Session

Contient les détails de la transaction à annuler.

Table des matières

Les membres de cette structure d'exception dépendent du contexte.

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

AbortTransactionResult

Service : Amazon QLDB Session

Contient les détails de la transaction abandonnée.

Table des matières

TimingInformation

Contient des informations sur les performances côté serveur pour la commande.

Type : objet TimingInformation

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

CommitTransactionRequest

Service : Amazon QLDB Session

Contient les détails de la transaction à valider.

Table des matières

CommitDigest

Spécifie le résumé de validation de la transaction à valider. Pour chaque transaction active, le résumé de validation doit être transmis. QLDB `CommitDigest` valide et rejette le commit avec une erreur si le condensé calculé sur le client ne correspond pas au résumé calculé par QLDB.

Le but de ce `CommitDigest` paramètre est de garantir que QLDB valide une transaction si et seulement si le serveur a traité l'ensemble exact d'instructions envoyées par le client, dans le même ordre que celui dans lequel le client les a envoyées, et sans doublons.

Type : objet de données binaires encodées en base64

Obligatoire : oui

TransactionId

Spécifie l'ID de transaction de la transaction à valider.

Type : chaîne

Contraintes de longueur : longueur fixe de 22.

Modèle : `^[A-Za-z-0-9]+$`

Obligatoire : oui

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

CommitTransactionResult

Service : Amazon QLDB Session

Contient les détails de la transaction validée.

Table des matières

CommitDigest

Le résumé de validation de la transaction validée.

Type : objet de données binaires encodées en base64

Obligatoire : non

ConsumedIOs

Contient des mesures relatives au nombre de demandes d'E/S consommées.

Type : objet IOUsage

Obligatoire : non

TimingInformation

Contient des informations sur les performances côté serveur pour la commande.

Type : objet TimingInformation

Obligatoire : non

TransactionId

L'ID de transaction de la transaction validée.

Type : chaîne

Contraintes de longueur : longueur fixe de 22.

Modèle : `^[A-Za-z-0-9]+$`

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

EndSessionRequest

Service : Amazon QLDB Session

Spécifie une demande de fin de session.

Table des matières

Les membres de cette structure d'exception dépendent du contexte.

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

EndSessionResult

Service : Amazon QLDB Session

Contient les détails de la session terminée.

Table des matières

TimingInformation

Contient des informations sur les performances côté serveur pour la commande.

Type : objet TimingInformation

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

ExecuteStatementRequest

Service : Amazon QLDB Session

Spécifie une demande d'exécution d'une instruction.

Table des matières

Statement

Spécifie l'énoncé de la demande.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 100 000

Obligatoire : oui

TransactionId

Spécifie l'ID de transaction de la demande.

Type : chaîne

Contraintes de longueur : longueur fixe de 22.

Modèle : `^[A-Za-z-0-9]+$`

Obligatoire : oui

Parameters

Spécifie les paramètres de l'instruction paramétrée dans la demande.

Type : tableau d’objets ValueHolder

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2

- AWS SDK pour Ruby V3

ExecuteStatementResult

Service : Amazon QLDB Session

Contient les détails de l'instruction exécutée.

Table des matières

ConsumedIOs

Contient des mesures relatives au nombre de demandes d'E/S consommées.

Type : objet IOUsage

Obligatoire : non

FirstPage

Contient les détails de la première page récupérée.

Type : objet Page

Obligatoire : non

TimingInformation

Contient des informations sur les performances côté serveur pour la commande.

Type : objet TimingInformation

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

FetchPageRequest

Service : Amazon QLDB Session

Spécifie les détails de la page à récupérer.

Table des matières

NextPageToken

Spécifie le jeton de page suivant de la page à récupérer.

Type : chaîne

Contraintes de longueur : longueur minimale de 4. Longueur maximum de 1024.

Modèle : `^[A-Za-z-0-9+/=]+$`

Obligatoire : oui

TransactionId

Spécifie l'ID de transaction de la page à récupérer.

Type : chaîne

Contraintes de longueur : longueur fixe de 22.

Modèle : `^[A-Za-z-0-9]+$`

Obligatoire : oui

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

FetchPageResult

Service : Amazon QLDB Session

Contient la page qui a été récupérée.

Table des matières

ConsumedIOs

Contient des mesures relatives au nombre de demandes d'E/S consommées.

Type : objet IOUsage

Obligatoire : non

Page

Contient les détails de la page récupérée.

Type : objet Page

Obligatoire : non

TimingInformation

Contient des informations sur les performances côté serveur pour la commande.

Type : objet TimingInformation

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

IOUsage

Service : Amazon QLDB Session

Contient les mesures d'utilisation des E/S pour une commande qui a été invoquée.

Table des matières

ReadIOs

Le nombre de demandes d'E/S de lecture effectuées par la commande.

Type : long

Obligatoire : non

WriteIOs

Le nombre de demandes d'E/S d'écriture effectuées par la commande.

Type : long

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

Page

Service : Amazon QLDB Session

Contient les détails de la page récupérée.

Table des matières

NextPageToken

Le jeton de la page suivante.

Type : chaîne

Contraintes de longueur : longueur minimale de 4. Longueur maximum de 1024.

Modèle : `^[A-Za-z-0-9+/=]+$`

Obligatoire : non

Values

Structure contenant des valeurs dans plusieurs formats de codage.

Type : tableau d'objets ValueHolder

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

StartSessionRequest

Service : Amazon QLDB Session

Spécifie une demande de démarrage d'une nouvelle session.

Table des matières

LedgerName

Le nom du registre sur lequel démarrer une nouvelle session.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Modèle : `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatoire : oui

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

StartSessionResult

Service : Amazon QLDB Session

Contient les détails de la session démarrée.

Table des matières

SessionToken

Jeton de session de la session démarrée. Cela `SessionToken` est obligatoire pour chaque commande suivante émise au cours de la session en cours.

Type : chaîne

Contraintes de longueur : longueur minimale de 4. Longueur maximum de 1024.

Modèle : `^[A-Za-z-0-9+/=]+$`

Obligatoire : non

TimingInformation

Contient des informations sur les performances côté serveur pour la commande.

Type : objet TimingInformation

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

StartTransactionRequest

Service : Amazon QLDB Session

Spécifie une demande pour démarrer une transaction.

Table des matières

Les membres de cette structure d'exception dépendent du contexte.

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

StartTransactionResult

Service : Amazon QLDB Session

Contient les détails de la transaction démarrée.

Table des matières

TimingInformation

Contient des informations de performance côté serveur pour la commande.

Type : objet TimingInformation

Obligatoire : non

TransactionId

ID de transaction de la transaction démarrée.

Type : chaîne

Contraintes de longueur : longueur fixe de 22.

Modèle : `^[A-Za-z-0-9]+$`

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

TimingInformation

Service : Amazon QLDB Session

Contient des informations de performance côté serveur pour une commande. Amazon QLDB capture les informations temporelles entre le moment où il reçoit la demande et le moment où il envoie la réponse correspondante.

Table des matières

ProcessingTimeMilliseconds

Le temps que QLDB a consacré au traitement de la commande, mesuré en millisecondes.

Type : long

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

ValueHolder

Service : Amazon QLDB Session

Structure qui peut contenir une valeur dans plusieurs formats de codage.

Table des matières

IonBinary

Une valeur binaire Amazon Ion contenue dans une `ValueHolder` structure.

Type : objet de données binaires encodées en base64

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 131072.

Obligatoire : non

IonText

Une valeur en texte brut Amazon Ion contenue dans une `ValueHolder` structure.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 1048576.

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

Erreurs courantes

Cette section répertorie les erreurs communes aux actions d'API de tous les services AWS. Pour les erreurs spécifiques à une action d'API pour ce service, consultez la rubrique pour cette action d'API.

AccessDeniedException

Vous ne disposez pas d'un accès suffisant pour effectuer cette action.

Code d'état HTTP : 400

IncompleteSignature

La signature de la requête n'est pas conforme aux normes AWS.

Code d'état HTTP : 400

InternalFailure

Le traitement de la demande a échoué en raison d'une erreur, d'une exception ou d'un échec inconnu.

Code d'état HTTP : 500

InvalidAction

L'action ou l'opération demandée n'est pas valide. Vérifiez que l'action est entrée correctement.

Code d'état HTTP : 400

InvalidClientTokenId

Le certificat X.509 ou l'ID de clé d'accès AWS fourni(e) n'existe pas dans nos archives.

Code d'état HTTP : 403

NotAuthorized

Vous ne disposez pas de l'autorisation nécessaire pour effectuer cette action.

Code d'état HTTP : 400

OptInRequired

L’ID de clé d’accès AWS a besoin d’un abonnement pour le service.

Code d'état HTTP : 403

RequestExpired

La demande a atteint le service plus de 15 minutes après la date affichée sur la demande ou plus de 15 minutes après la date d'expiration de la demande (comme pour les URL pré-signées) ou la date affichée sur la demande est postérieure de 15 minutes.

Code d'état HTTP : 400

ServiceUnavailable

La requête a échoué en raison d'une défaillance temporaire du serveur.

HTTP Status Code: 503

ThrottlingException

La demande a été refusée suite à une limitation des demandes.

Code d'état HTTP : 400

ValidationError

L'entrée ne satisfait pas les contraintes spécifiées par un service AWS.

Code d'état HTTP : 400

Paramètres communs

La liste suivante contient les paramètres que toutes les actions utilisent pour signer les demandes Signature Version 4 à l'aide d'une chaîne de requête. Tous les paramètres spécifiques d'une action particulière sont énumérées dans le sujet consacré à cette action. Pour plus d'informations sur Signature version 4, consultez la section Signature de demandes d'AWSAPI dans le Guide de l'utilisateur IAM.

Action

Action à effectuer.

Type : chaîne

Obligatoire : oui

Version

Version de l'API pour laquelle la demande est écrite, au format AAAA-MM-JJ.

Type : chaîne

Obligatoire : oui

X-Amz-Algorithm

Algorithme de hachage que vous avez utilisé pour créer la signature de la demande.

Condition : spécifiez ce paramètre lorsque vous incluez des informations d'authentification dans une chaîne de requête plutôt que dans l'en-tête d'autorisation HTTP.

Type : chaîne

Valeurs valides : `AWS4-HMAC-SHA256`

Obligatoire : Conditionnelle

X-Amz-Credential

Valeur de la portée des informations d'identification, qui est une chaîne incluant votre clé d'accès, la date, la région cible, le service demandé et une chaîne de terminaison (« aws4_request »). Spécifiez la valeur au format suivant : access_key/AAAAMMJJ/région/service/aws4_request.

Pour plus d'informations, consultez la section Création d'une demande d'AWSAPI signée dans le Guide de l'utilisateur IAM.

Condition : spécifiez ce paramètre lorsque vous incluez des informations d'authentification dans une chaîne de requête plutôt que dans l'en-tête d'autorisation HTTP.

Type : chaîne

Obligatoire : Conditionnelle

X-Amz-Date

La date utilisée pour créer la signature. Le format doit être au format de base ISO 8601 (AAAAMMJJ'T'HHMMSS'Z'). Par exemple, la date/heure suivante est une valeur X-Amz-Date valide : `20120325T120000Z`.

Condition : X-Amz-Date est un en-tête facultatif pour toutes les demandes. Il peut être utilisé pour remplacer la date dans la signature des demandes. Si l'en-tête Date est spécifié au format de base ISO 8601, X-Amz-Date n'est pas obligatoire. Lorsque X-Amz-Date est utilisé, il remplace toujours la valeur de l'en-tête Date. Pour plus d'informations, consultez la section Éléments d'une signature de demande d'AWSAPI dans le Guide de l'utilisateur IAM.

Type : chaîne

Obligatoire : Conditionnelle

X-Amz-Security-Token

Le jeton de sécurité temporaire obtenu lors d'un appel àAWS Security Token Service (AWS STS). Pour obtenir la liste des services prenant en charge les informations d'identification de sécurité temporaires deAWS STS, consultez la section Services AWSqui fonctionnent avec IAM dans le Guide de l'utilisateur IAM.

Condition : si vous utilisez des informations d'identification de sécurité temporaires deAWS STS, vous devez inclure le jeton de sécurité.

Type : chaîne

Obligatoire : Conditionnelle

X-Amz-Signature

Spécifie la signature codée en hexadécimal qui a été calculée à partir de la chaîne à signer et de la clé de signature dérivée.

Condition : spécifiez ce paramètre lorsque vous incluez des informations d'authentification dans une chaîne de requête plutôt que dans l'en-tête d'autorisation HTTP.

Type : chaîne

Obligatoire : Conditionnelle

X-Amz-SignedHeaders

Spécifie tous les en-têtes HTTP qui ont été inclus dans la demande canonique. Pour plus d'informations sur la spécification d'en-têtes signés, consultez la section Création d'une demande d'AWSAPI signée dans le Guide de l'utilisateur IAM.

Condition : spécifiez ce paramètre lorsque vous incluez des informations d'authentification dans une chaîne de requête plutôt que dans l'en-tête d'autorisation HTTP.

Type : chaîne

Obligatoire : Conditionnelle

Quotas et limites d'Amazon QLDB

Cette section décrit les quotas actuels, également appelés limites, dans Amazon QLDB.

Rubriques

Quotas par défaut

QLDB possède les quotas par défaut suivants, qui sont également répertoriés dans les points de terminaison Amazon QLDB et les quotas dans le Références générales AWS. Ces quotas sontCompte AWS par région. Pour demander une augmentation de quota pour votre compte dans une région, vous pouvez utiliser la console Service Quotas.

Connectez-vous à laAWS Management Console et ouvrez la console Service Quotas à l'adresse : https://console.aws.amazon.com/servicequotas/.

Ressource	Quota par défaut
Nombre maximal de registres actifs que vous pouvez créer pour ce compte dans la région actuelle	5
Nombre maximum d'exportations de journaux actifs vers Amazon S3 par registre	2
Nombre maximum de flux de journal actifs vers Kinesis Data Streams par registre	5

Quotas fixes

Outre les quotas par défaut, QLDB dispose des quotas fixes suivants par registre. Ces quotas ne peuvent pas être augmentés à l'aide Service Quotas :

Ressource	Quotas fixe
Nombre de sessions actives simultanées	1 500
Nombre de tables actives	20
Nombre total de tables (actives et inactives) **Note** Dans QLDB, les tables supprimées sont considérées comme inactives et prises en compte dans ce quota total.	40
Nombre d'index par table	5
Nombre de documents inclus dans une transaction	40
Nombre de révisions à supprimer dans une transaction	1
Taille du document (encodé dans le `IonBinary` format)	128 Ko
Taille des paramètres de l'instruction (`IonBinary` format)	128 Ko
Taille des paramètres de l'instruction (`IonText` format)	1 Mo
Longueur de chaîne d'instruction	100 000 caractères
Taille de la transaction	4 Mo

Ressource	Quotas fixe
Délai d'expiration de transaction	30 secondes
Période d'expiration pour les tâches d'exportation de journaux terminées	7 jours
Période d'expiration pour les flux de journaux du terminal	7 jours

Quotas de registre

Pour demander une augmentation de quota comptable pour votre compte dans une région, vous pouvez utiliser la console Service Quotas.

Ouvrez la console Service Quotas à l'adresse https://console.aws.amazon.com/servicequotas/.

Certains cas d'utilisation de QLDB nécessitent un nombre croissant de registresCompte AWS par région en fonction de la croissance de l'activité. Par exemple, vous devrez peut-être créer des registres dédiés pour isoler les clients ou les données. Dans ce cas, envisagez de tirer parti d'une architecture multi-comptes pour travailler avec les quotas QLDB. Pour plus d'informations, consultez la section Isolation des silos de comptes dans leAWS livre blanc sur les stratégies d'isolation des locataires SaaS.

Taille des documents

La taille maximale d'un document codé dans ce`IonBinary` format est de 128 Ko. Nous ne pouvons pas fournir de limite exacte pour la taille d'un document au`IonText` format, car la conversion du texte au format binaire varie considérablement en fonction de la structure de chaque document. QLDB prend en charge les documents à contenu ouvert, de sorte que chaque structure de document unique modifie le calcul de la taille.

Taille de la transaction

La taille maximale d'une transaction dans QLDB est de 4 Mo. La taille d'une transaction est calculée sur la base de la somme des facteurs suivants.

Deltas

Les modifications du document générées par tous les relevés de la transaction. Dans le cas d'une transaction qui concerne plusieurs documents, la taille totale du delta est la somme du delta individuel de chaque document concerné.

Metadonnées

Les métadonnées de transaction générées par le système qui sont associées à chaque document concerné.

Index

Si un index est défini sur une table affectée par la transaction, l'entrée d'index associée génère également un delta.

Historique

Comme toutes les révisions de documents sont conservées dans QLDB, toutes les transactions sont également ajoutées à l'historique.

Insertions : chaque document inséré dans un tableau possède également une copie insérée dans son tableau historique. Par exemple, un document de 100 Ko nouvellement inséré génère un minimum de 200 Ko de deltas dans une transaction. (Il s'agit d'une estimation approximative qui n'inclut pas les métadonnées ni les index.)

Mises à jour : toute mise à jour du document, même pour un seul champ, crée une nouvelle révision de l'ensemble du document dans l'historique, plus ou moins le delta de la mise à jour. Cela signifie qu'une petite mise à jour d'un document volumineux générerait tout de même un delta de transaction important. Par exemple, l'ajout de 2 Ko de données dans un document existant de 100 Ko crée une nouvelle révision de 102 Ko dans l'historique. Cela représente au moins 104 Ko de deltas totaux dans une transaction. (Encore une fois, cette estimation n'inclut pas les métadonnées ni les index.)

Suppressions : à l'instar des mises à jour, toute transaction de suppression crée une nouvelle révision du document dans l'historique. Toutefois, la`DELETE` révision nouvellement créée est plus petite que le document d'origine car elle contient des données utilisateur nulles et ne contient que des métadonnées.

Contraintes d'affectation de noms

Le tableau ci-dessous décrit les contraintes d'affectation de noms dans Amazon QLDB.

Nom du registre	• Ne doit contenir que entre 1 et 32 caractères alphanumériques ou traits d'union. • Le premier et le dernier caractère doivent être composés d'une lettre ou d'un chiffre. • Il ne doit pas y avoir que des chiffres. • Ils ne peuvent pas comporter deux traits d'union consécutifs. • Il est sensible à la casse.
Nom du flux de journal	
Nom de la table	• Ne doit contenir que entre 1 et 128 caractères alphanumériques ou traits de soulignement. • Doit contenir une lettre ou un trait de soulignement pour le premier caractère. • Peut contenir n'importe quelle combinaison de caractères alphanumériques et de traits de soulignem ent pour les autres caractères. • Il est sensible à la casse. • Il ne doit pas s'agir d'un mot réservé QLDB PartiQL.

QLDB à

Les ressources connexes suivantes peuvent s'avérer utiles lors de l'utilisation de ce service.

Rubriques

- Documentation technique
- GitHub référentiels
- AWSarticles et articles de blog
- Multimédia
- Ressources AWS générales

Documentation technique

- FAQ Amazon QLDB : questions fréquemment posées sur le produit.
- Tarification Amazon QLDB : informations sur lesAWS tarifs et exemples.
- AWS re:Post— forumAWS communautaire pour les questions et réponses (Q&A).
- Amazon Ion : guides du développeur, guides d'utilisation et références pour le format de données Amazon Ion.
- PartiQL — Document de spécification et didacticiels généraux pour le langage de requête PartiQL.
- Ateliers QLDB : ateliers proposant des exemples pratiques d'utilisation d'Amazon QLDB pour créer system-of-record des applications, notamment les ateliers suivants :
  - Apprde données de données de données de données de
  - Utilisation d'Amazon Ion et conversion d'Ion vers et depuis JSON (Java)
  - UtilisationAWS Glue d'Amazon Athena pour activer les données QLDB pour un lac de données
  - Fde données de données de données de données de données de données de données de Amazon Aurora de MySQL de données de données
- Des données de qualité inviolables à l'aide d'Amazon QLDB : une implémentation deAWS solutions qui montre comment empêcher les attaquants de falsifier des données de qualité en utilisant QLDB pour conserver un historique précis des modifications de données. AWS Les implémentations de solutions vous aident à résoudre les problèmes courants et à les utiliser plus rapidementAWS.

GitHub référentiels

Pilotes

- Pilote .NET : implémentation .NET du pilote QLDB.
- Go driver : implémentation Go du pilote QLDB.
- Pilote Java : implémentation Java du pilote QLDB.
- Pilote Node.js : implémentation Node.js du pilote QLDB.
- Pilote Python : implémentation Python du pilote QLDB.

interface de commande

- QLDB shell : implémentation Python et Rust d'une interface de ligne de commande pour l'API de données transactionnelles QLDB.

Exemples d'applications

- Application Java DMV : application didacticiel basée sur un cas d'utilisation du ministère des véhicules automobiles (DMV). Il présente les opérations de base et les meilleures pratiques d'utilisation de QLDB et du pilote QLDB pour Java.
- Application .NET DMV : application de didacticiel basée sur DMV qui présente les opérations de base et les meilleures pratiques relatives à l'utilisation de QLDB et du pilote QLDB pour .NET.
- Application DMV Node.js : application de didacticiel basée sur DMV qui présente les opérations de base et les meilleures pratiques relatives à l'utilisation de QLDB et du pilote QLDB pour Node.js.
- Application DMV Python : application de didacticiel basée sur DMV qui présente les opérations de base et les meilleures pratiques relatives à l'utilisation de QLDB et du pilote QLDB pour Python.
- Ledger loader : framework Java pour le chargement asynchrone de données dans un registre QLDB à grande vitesse à l'aide d'un canal de diffusion compatible (AWS DMSAmazon SQS, Amazon SNS, Kinesis Data Streams, Amazon MSK ou EventBridge).
- Processeur d'exportation : framework Java extensible qui gère le traitement des exportations QLDB dans Amazon S3 en lisant les résultats de l'exportation et en parcourant les blocs exportés en séquence.
- Exemple de flux QLDB Lambda en Python : application qui montre comment consommer des flux QLDB à l'aide d'uneAWS Lambda fonction permettant d'envoyer des données QLDB à une rubrique Amazon SNS à laquelle une file d'attente Amazon SQS est abonnée.

- Exemple OpenSearch d'intégration de flux QLDB : application Python qui montre comment intégrer Amazon OpenSearch Service à QLDB à l'aide de flux.
- Application à double entrée : application Java qui montre comment modéliser une application de registre financier à double entrée à l'aide de QLDB.
- QLDB KVS pour Node.js — Une bibliothèque d'interface de stockage clé-valeur simple pour QLDB avec des fonctions supplémentaires pour la vérification des documents.

SerDe

- Bibliothèques Amazon Ion : bibliothèques, outils et documentation pris en charge par l'équipe Ion.
- Implémentation de PartiQL : implémentation, spécification et didacticiels pour PartiQL.

AWSarticles et articles de blog

- Comment Earnin a développé son service de registre à l'aide d'Amazon QLDB (16 février 2023) — Décrit comment Earnin a utilisé QLDB pour créer un service de registre afin de fournir à ses utilisateurs une application financière mobile moderne et complète.
- Exportez et analysez les données des journaux Amazon QLDB à l'aideAWS Glue d'Amazon Athena (19 décembre 2022) — Explique comment vous pouvez utiliser la fonctionnalité d'exportation de QLDB avecAWS Glue et Athena pour fournir des fonctionnalités de reporting et d'analyse à vos architectures basées sur des registres.
- Comment Shinsegae International améliore l'expérience client et prévient la contrefaçon grâce à Amazon QLDB (3 août 2022) — Décrit comment Shinsegae International a créé un service de vérification de l'authenticité numérique utilisant Amazon QLDB pour informer les clients de l'authenticité des produits de luxe et fournir l'historique des achats et de la distribution des produits.
- BungKusit utilise Amazon QLDB et la technologie ISV de VeriDoc Global pour améliorer l'expérience des clients et des agents de livraison (29 avril 2022) — Montre comment une entreprise de logistique, BungKusit, a utilisé QLDB pour mettre en œuvre une plateforme centralisée et transparente de communication intersectorielle avec un journal de transactions immuable et vérifiable par cryptographie.
- Utiliser Amazon QLDB en tant que magasin clé-valeur immuable avec une API REST et un JSON (14 février 2022) — Présentation d'un moyen de travailler avec QLDB en tant que magasin clé-valeur immuable et d'utiliser ses fonctionnalités d'audit via une API REST.

- Création d'un système bancaire de base avec Amazon QLDB (21 janvier 2022) — Explique comment utiliser QLDB pour créer un système de registre bancaire de base. Cela montre également pourquoi QLDB est une bonne fit-for-purpose base de données qui répond aux besoins du secteur des services financiers.
- Comment Specright utilise Amazon QLDB pour créer un réseau de chaîne d'approvisionnement traçable (21 janvier 2022) — Explique comment Specright utilise QLDB pour créer un réseau de chaîne d'approvisionnement traçable qui permet aux marques grossistes, aux détaillants et aux fabricants de partager des données critiques sur la chaîne d'approvisionnement et des données sur les spécifications d'emballage.
- Comment FeMR fournit des données médicales cryptographiquement sécurisées et vérifiables avec Amazon QLDB (23 décembre 2021) — Explore comment Team FeMR a utilisé QLDB et d'autres servicesAWS gérés pour soutenir ses efforts de secours.
- Surveiller les modèles d'accès aux requêtes Amazon QLDB (8 novembre 2021) : montre comment associer les transactions QLDB et les instructions partiQL exécutées dans les transactions aux demandes d'API reçues via Amazon API Gateway.
- Créez une opération CRUD et un flux de données simples sur Amazon QLDB à l'aide deAWS Lambda (28 septembre 2021) : montre comment effectuer des opérations CRUD (création, lecture, mise à jour et suppression) sur QLDB à l'aide deAWS Lambda fonctions.
- Vérification cryptographique en conditions réelles avec Amazon QLDB (26 août 2021) — Explique la valeur de la vérification cryptographique dans un registre QLDB dans le contexte d'un cas d'utilisation réaliste.
- Vérifier les conditions de livraison avec le projet Accord et Amazon QLDB : partie 1, partie 2 (28 juin 2021) — Explique comment appliquer la technologie des contrats juridiques intelligents pour vérifier les conditions de livraison avec le projet Accord et QLDB open source.
- Diffusion de données Amazon QLDB viaAWS CDK (7 juin 2021) : montre comment utiliser leAWS Cloud Development Kit (AWS CDK) pour configurer QLDB, renseigner les données QLDB à l'aide deAWS Lambda fonctions et configurer le streaming QLDB afin de garantir la résilience des données du registre.
- Démonstration du contrôle d'accès détaillé dans QLDB (1er juin 2021) — Indique comment démarrer avec des autorisations détailléesAWS Identity and Access Management (IAM) pour un registre QLDB.
- Traitement des flux avec les DynamoDB Streams et les flux QLDB (23 novembre 2020) — Fournit une brève comparaison entre les flux DynamoDB et les flux QLDB, ainsi que des conseils pour démarrer avec ces flux.

- Diffusion de données depuis Amazon QLDB vers OpenSearch (19 août 2020) — Décrit comment diffuser des données depuis QLDB vers Amazon OpenSearch Service afin de prendre en charge la recherche en texte enrichi et les analyses en aval, telles que l'agrégation ou les statistiques entre les enregistrements.
- Comment j'ai diffusé des données d'Amazon QLDB vers DynamoDB à l'aide de Nodejs en temps quasi réel (7 juillet 2020) — Décrit comment diffuser des données de QLDB vers DynamoDB afin de prendre en charge une latence à un chiffre et des requêtes clé-valeur évolutives à l'infini.
- Création d'une interface GraphQL pour Amazon QLDB avecAWS AppSync : Partie 1 , Partie 2 (4 mai 2020) — Explique comment intégrer QLDB etAWS AppSync fournir une API polyvalente alimentée par GraphQL en plus d'un registre QLDB.

Multimédia

Vidéos AWS

- AWSTutoriels et démos : Streaming de QLDB vers Aurora (17 mars 2023 ; 21 minutes) — Cette vidéo explique les concepts fondamentaux pour implémenter la fonctionnalité de streaming QLDB et montre comment configurer le streaming de données depuis QLDB vers une base de données Amazon Aurora MySQL en aval.
- ArcBlock: Tirer parti d'Amazon QLDB pour créer une solution d'identité décentralisée (31 mai 2022 ; 4 minutes) : présente ArcBlock la solution d'identité décentralisée qu'ils ont créée à l'aide de QLDB.
- AWSre:Invent 2020 : Utilisation d'Amazon QLDB comme system-of-trust base de données pour les applications professionnelles principales (5 février 2021 ; 30 minutes) — Découvrez comment les premiers utilisateurs d'Amazon QLDB ont appliqué les propriétés uniques de la base de données Ledger en matière de provenance des données et de vérifiabilité cryptographique afin de mettre en œuvre des systèmes d'enregistrement intégrant l'intégrité des données.
- AWSre:Invent 2020 : Création d'une application sans serveur avec Amazon QLDB (5 février 2021 ; 28 minutes) — Découvrez comment créer, tester et optimiser une application sans serveur entièrement fonctionnelle en combinant Amazon QLDB avec des services telsAWS Lambda qu'Amazon Kinesis et Amazon DynamoDB.
- AWSre:Invent 2020 : Création d'applications basées sur des audits qui préservent l'intégrité des données avec Amazon QLDB (5 février 2021 ; 18 minutes) — Cette session explore les problèmes qu'Amazon QLDB peut résoudre, répond à vos questions sur les raisons et les raisons pour

lesquelles vous devriez utiliser une base de données grand livre, et partage des cas d'utilisation de clients tels qu'Osano.

- Comment stocker de manière centralisée des journaux immuables à l'aide d'Amazon QLDB avec des applications .NET (7 décembre 2020 ; 10 minutes) — Démonstration de l'utilisation de QLDB avec des applications .NET pour stocker de manière centralisée des journaux immuables.
- Atelier virtuel : Création de systèmes d'enregistrement basés sur des registres avec QLDB et Java - Discussions techniquesAWS en ligne (29 juillet 2020 ; 87 minutes) — Atelier animé par un instructeur qui présente le laboratoire Working With Ion Immersion Day afin de créer un programme de chargement de données utilisant la bibliothèque Amazon Ion et le pilote QLDB pour Java.
- Atelier pour développeurs : UtilisationAWS de la base de données QLDB immuable sur ABT Node (22 juin 2020 ; 34 minutes) — Un step-by-step guide sur la façon de configurer et de configurer QLDB sur ABT Node. Il explique également comment installer et configurer ArcBlock les blocklets Blockchain Explorer et Boarding Gate pour se connecter à QLDB.
- AWSConférence technique : le point de vue d'un client sur la création d'une application de système d'enregistrement déclenchée par un événement avec Amazon QLDB (31 mars 2020 ; 29 minutes) — Une conférence technique de Matt Lewis, héros desAWS données et architecte en chef de la Driver and Vehicle Licensing Agency (DVLA) au Royaume-Uni, qui explique le cas d'utilisation de QLDB et de l'architecture événementielle de son application.
- AWSre:Invent 2019 : Pourquoi vous avez besoin d'une base de données comptable : BMW, DVLA et Sage discutent de cas d'utilisation (5 décembre 2019 ; 47 minutes) — Une présentation des clients BMW, l'organisation gouvernementale britannique DVLA et Sage qui explique les raisons d'utiliser une base de données de registres et partage leurs cas d'utilisation pour QLDB.
- AWSre:Invent 2019 : Amazon QLDB : analyse approfondie par un ingénieur des raisons pour lesquelles cela change la donne (5 décembre 2019 ; 50 minutes) — Une présentation d'Andrew Certain (ingénieurAWS distingué) qui aborde l'architecture unique de QLDB, inédite dans un journal, ainsi que ses différentes innovations. Il inclut le hachage cryptographique, les arbres Merkle, la réplication de zones de disponibilité multiples et la prise en charge de PartiQL.
- Création d'applications avec Amazon QLDB, une base de données de registre unique en son genre - Discussions techniquesAWS en ligne (19 novembre 2019 ; 51 minutes) — Une conférence technique approfondie qui décrit les fonctionnalités uniques de QLDB et explique en détail comment utiliser les fonctionnalités de base. Cela inclut l'interrogation de l'historique complet de vos données, la vérification cryptographique des documents et la conception d'un modèle de données.

Podcasts

- Pourquoi les clients choisissent-ils Amazon QLDB ? (5 juillet 2020 ; 33 minutes) — Une discussion qui explique la définition d'une base de données grand livre, en quoi elle est différente d'une blockchain et comment les clients l'utilisent aujourd'hui.

Ressources AWS générales

- Formations et ateliers : liens vers des formations spécialisées et basées sur les rôles, ainsi que des ateliers d'autoformation pour améliorer vos compétences AWS et acquérir une expérience pratique.
- Centre pour développeurs AWS : parcourez des didacticiels, téléchargez des outils et découvrez les événements pour les développeurs AWS.
- Outils pour développeur AWS : liens vers des outils pour développeur, kits SDK, boîtes à outils IDE et outils de ligne de commande pour développer et gérer des applications AWS.
- Centre de ressources pour la mise en route : découvrez comment configurer votre Compte AWS, rejoindre la communauté AWS et lancer votre première application.
- Tuétape par étape par step-by-step étape par étape pour lancer votre première application surAWS.
- Livres blancs AWS : liens vers une liste complète des livres blancs techniques AWS couvrant des sujets tels que l'architecture, la sécurité et l'économie, créés par des architectes de solutions AWS ou d'autres experts techniques.
- AWS SupportCentre – Hub pour la création et la gestion de vos cas AWS Support. Inclut également des liens vers d'autres ressources utiles, telles que des forums, des FAQ techniques, l'état de santé d'un service et AWS Trusted Advisor.
- AWS Support— Principale page web d'informations à propos d'AWS Support one-on-one, un canal d'assistance technique rapide pour vous aider à développer et à exécuter des applications dans le cloud.
- Contactez-nous : point de contact central pour toute question relative à la facturation AWS, à votre compte, aux événements, à des abus ou à d'autres problèmes.
- AWSConditions d'utilisation du site : informations détaillées sur nos droits d'auteur et notre marque, sur votre compte, votre licence et votre accès au site, et sur d'autres sujets.

Historique de versions pour Amazon QLDB

Le tableau ci-après décrit les modifications importantes apportées dans chaque version Amazon QLDB et les mises à jour correspondantes dans le Manuel du développeur Amazon QLDB. Pour recevoir les notifications sur les mises à jour de cette documentation, vous pouvez vous abonner au Flux RSS.

- Version de l'API : 2019-01-02
- Date de la dernière mise à jour de la documentation : 3 janvier 2023

Modification	Description	Date
Directives IAM mises à jour	Guide mis à jour pour s'aligner sur les bonnes pratiques IAM. Pour de plus amples informations, veuillez consulter Bonnes pratiques de sécurité dans IAM.	3 janvier 2023
Mise à jour des politiquesAWS gérées	Amazon QLDB a mis à jour les politiquesAWS gérées existantes `AmazonQLDBFullAccess` et `AmazonQLDBConsoleFullAccess`. Ces politiques disposent d'une nouvelle autorisation qui permet aux responsables de rédiger des révisions de documents à l'aide d'une procédure stockée dans PartiQL. Pour plus d'informations, consultez les politiquesAWS gérées pour Amazon QLDB.	4 novembre 2022

Rédaction de données	Amazon QLDB prend désormais en charge la procédure stockée`REDACT_REVISION` PartiQL pour les registres créés le 22 juillet 2021 ou après cette date. À l'aide de cette procédure stockée, vous pouvez supprimer définitivement les révisions de documents inactives dans l'historique tout en préservant l'intégrité globale des données de votre registre. Pour de plus amples informations, veuillez consulter Rédaction d'instructions.	3 novembre 2022
Pilote Node.js v3	Le pilote Amazon QLDB pour Node.js version 3.0 est désormais disponible pour tous. Cette version introduit le support pour laAWS SDK for JavaScript v3. Pour les notes de publication, consultez le GitHub référentiel awslabs/amazon-qldb-driver-nodejs.	26 septembre 2022
Go Driver v3	Le pilote Amazon QLDB pour Go version 3.0 est désormais disponible pour tous. Cette version introduit le support pour laAWS SDK for Go v2. Pour les notes de publication, consultez le GitHub référentiel awslabs/amazon-qldb-driver-go.	11 août 2022

Nouvel éditeur de requêtes PartiQL	Un nouvel éditeur de requêtes PartiQL sur la console Amazon QLDB est désormais disponible de manière générale. Le nouvel éditeur QLDB PartiQL fournit une interface améliorée pour créer des requêtes, déboguer des transactions et explorer les résultats. Pour plus d'informations sur l'ouverture et l'utilisation de l'éditeur, consultez Accès à Amazon QLDB à l'aide de la console.	22 juin 2022
Mappeur d'objets Ion pour pilote .NET	Le pilote Amazon QLDB pour .NET version 1.3 prend en charge le mappeur d'objets Ion. Cette fonctionnalité vous permet d'éviter complètement de devoir effectuer une conversion manuelle entre les types Amazon Ion et les types C# natifs. Pour consulter l'historique complet des modifications du mappeur d'objets Ion, consultez le fichier CHANGELOG.md dans le GitHub référentiel `amzn/ion-object-mapper-dotnet` .	19 janvier 2022

Format d'exportation du journal JSON	Amazon QLDB prend désormais en charge le format de sortie JSON Lines pour les exportations de journaux. Pour plus d'informations, consultez Exportation de données de journaux depuis Amazon QLDB.	21 décembre 2021
Dépannage d'Amazon MQ	Ajout d'une nouvelle rubrique de résolution des problèmes qui fournit des conseils pour une liste agrégée des erreurs courantes que vous pouvez rencontrer lors de l'utilisation d'Amazon QLDB.	8 décembre 2021
Lancement d'une nouvelle région	Amazon QLDB est désormais disponible dans la région Canada (Centre). Pour obtenir la liste complète des régions disponibles, consultez les points de terminaison et les quotas Amazon QLDB dans le Référence générale d'Amazon Web Services.	11 novembre 2021

Prévention du problème de l'adjoint confus entre services	Amazon QLDB prend désormais en charge l'utilisation `aws:SourceArn` des clés de contexte de condition `aws:SourceAccount` globale dans les politiques de ressources IAM afin d'empêcher le problème de l'adjoint confus. Pour plus d'informations, consultez Prévention du problème de l'adjoint confus entre services.	8 novembre 2021
Coque Amazon QLDB v2	La version 2.0 du shell Amazon QLDB, qui est écrit en Rust, est désormais généralement disponible. Pour les notes de publication, consultez le GitHub référentiel awslabs/amazon-qldb-shell.	14 octobre 2021

Mise à jour des politiquesAWS gérées	Amazon QLDB a mis à jour les politiques AWS gérées existantes `AmazonQLDBFullAccess` et `AmazonQLDBConsoleFullAccess` . Ces politiques comportent de nouvelles autorisations permettant aux responsables de transmettre n'importe quelle ressource de rôle IAM de votre compte au service QLDB. Cela est obligatoire pour toutes les demandes d'exportation et de flux de journal. Pour plus d'informations, consultez les politiquesAWS gérées pour Amazon QLDB.	2 septembre 2021
AWS KMSClés gérées par le client	Amazon QLDB prend désormais en charge le chiffrement au repos à l'aide de clés gérées par le client dansAWS Key Management Service (AWS KMS) pour les nouvelles ressources du registre. Pour de plus amples informations, veuillez consulter Chiffrement au repos dans Amazon QLDB.	22 juillet 2021

Mise à jour de la politiqueAWS gérée	Amazon QLDB a mis à jour la politiqueAWS gérée existante`AmazonQLDBReadOnly` afin de supprimer une`qldb:GetBlock` action dupliquée qui était précédemment répertoriée deux fois. Pour plus d'informations, consultez les politiquesAWS gérées pour Amazon QLDB.	1er juillet 2021
Lancement d'une nouvelle région	Amazon QLDB est désormais disponible dans la région Europe (London). Pour obtenir la liste complète des régions disponibles, consultez les points de terminaison et les quotas Amazon QLDB dans le Référence générale d'Amazon Web Services.	24 juin 2021

Mise à jour des politiquesAWS gérées	Amazon QLDB a mis à jour les politiques AWS gérées existantes `AmazonQLDBFullAccess` et `AmazonQLDBConsoleFullAccess` . Ces politiques comportent de nouvelles autorisations qui permettent aux responsables de mettre à jour le mode d'autorisations dans tous les registres et d'exécuter toutes les commandes PartiQL dans tous les registres d'`STANDARD`autorisations. Pour plus d'informations, consultez les politiquesAWS gérées pour Amazon QLDB.	27 mai 2021
Mode d'autorisations standard	Amazon QLDB prend désormais en charge un mode d'`STANDARD`autorisations pour les ressources du registre. Le mode d'autorisations standard, vous pouvez contrôler l'accès avec une granularité plus fine pour les registres, les tables et les commandes PartiQL. Pour plus d'informations, consultez Premiers pas avec le mode d'autorisations standard.	27 mai 2021

Statistiques partielles sur les instructions SQL	La fonctionnalité de statistiques des instructions partiQL est désormais disponible dans l'éditeur de requêtes de la console Amazon QLDB. Pour de plus amples informations, veuillez consulter Obtenir des statistiques pour les instructions.	24 mai 2021
Tutoriel : Vérification des données à l'aide d'unAWS SDK	Ajout d'un step-by-step didacticiel contenant des exemples de code qui montrent comment vérifier un hachage de révision et un hachage de bloc dans Amazon QLDB à l'aide de l'API QLDB via unAWS SDK. Pour en savoir plus, consultez le didacticiel : vérification des données à l'aide d'unAWS SDK.	6 mai 2021
Pilote .NET v1.2	Le pilote Amazon QLDB pour .NET version 1.2 est désormais disponible pour tous. Cette version introduit des API asynchrones. Pour les notes de publication, consultez le GitHub référentiel awslabs/amazon-qldb-driver-dotnet.	1 avril 2021

Instruction PartiQL`DROP INDEX`	Dans Amazon QLDB, PartiQL prend désormais en charge l'instruction DROP INDEX. Pour de plus amples informations, veuillez consulter Suppression d'index.	3 mars 2021
Statistiques partielles sur les instructions SQL	La fonctionnalité de statistiques des instructions partiQL est désormais disponible dans la dernière version du pilote Amazon QLDB pour tous les langages pris en charge, notamment .NET, Go et Python. Pour de plus amples informations, veuillez consulter Obtenir des statistiques pour les instructions.	25 février 2021
TypeScript tutoriel de démarrage rapide	Exemples de TypeScript code ajoutés dans le didacticiel de démarrage rapide pour le pilote Amazon QLDB pour Node.js.	28 décembre 2020
Statistiques partielles sur les instructions SQL	La dernière version du pilote Amazon QLDB pour Java et Node.js fournit désormais des statistiques d'exécution des instructions qui peuvent vous aider à exécuter des instructions PartiQL plus efficaces. Pour de plus amples informations, veuillez consulter Obtenir des statistiques pour les instructions.	22 décembre 2020

Références de livres de recettes et didacticiels de démarrage rapide pour les pilotes	Ajout de références de livres de recettes pour les pilotes Go et Node.js, de didacticiels de démarrage rapide pour les pilotes Java et Python et d'un guide d'utilisation d'Amazon Ion dans QLDB. Pour plus d'informations, consultez Premiers pas avec le pilote Amazon QLDB.	24 novembre 2020
Coque Amazon QLDB v1.1	La version 1.1 du shell Amazon QLDB est désormais disponible pour le grand public. Pour les notes de publication, consultez le GitHub référentiel awslabs/amazon-qldb-shell.	26 octobre 2020
Pilote Amazon QLDB pour Go	Le pilote Amazon QLDB pour Go d'Amazon QLDB pour Go est désormais généralement disponible. Ce pilote est open source GitHub et vous permet de l'utiliserAWS SDK for Go pour interagir avec l'API de données transactionnelles de QLDB. Pour les notes de publication, consultez le GitHub référentiel awslabs/amazon-qldb-driver-go.	20 octobre 2020

Recommandations de configuration du pilote Node.js	Ajout d'une nouvelle section qui fournit des recommandations de configuration pour le pilote Amazon QLDB pour Node.js, notamment comment réduire la latence en réutilisant les connexions avec keep-alive.	16 octobre 2020
Création d'index sur des tables non vides	Amazon QLDB prend désormais en charge la création d'index sur des tables non vides. Pour de plus amples informations, veuillez consulter Gestion des index.	30 septembre 2020
Optimisation des performances des requêtes	Ajout d'une nouvelle section qui décrit les contraintes de requête dans Amazon QLDB et fournit des conseils pour optimiser les performances des requêtes compte tenu de ces contraintes.	18 septembre 2019
Guide de référence pour le pilote Java	Ajout d'une référence au livre de recettes qui présente des exemples de code de cas d'utilisation courants pour le pilote Amazon QLDB pour Java.	3 septembre 2020

Gestion des sessions avec le pilote	Ajout d'une nouvelle section qui donne une vue d'ensemble de la gestion des sessions avec le pilote dans Amazon QLDB et décrit la manière dont le pilote QLDB gère les sessions lors de l'exécution de transactions de données.	1 septembre 2020
Pilote Java v2.0	Le pilote Amazon QLDB pour Java version 2.0 est désormais disponible pour tous. Pour les notes de publication, consultez le GitHub référentiel awslabs/amazon-qldb-driver-java.	28 août 2020
Pilote Node.js v2.0	Le pilote Amazon QLDB pour Node.js version 2.0 est désormais disponible pour tous. Pour les notes de publication, consultez le GitHub référentiel awslabs/amazon-qldb-driver-nodejs.	27 août 2020
Informations connexes	Ajout d'une nouvelle rubrique contenant des liens vers des informations connexes et des ressources supplémentaires pour vous aider à comprendre et à utiliser Amazon QLDB.	24 août 2020

Pilote Python v3.0	Le pilote Amazon QLDB pour Python version 3.0 est désormais disponible pour tous. Pour les notes de publication, consultez le GitHub référentiel awslabs/amazon-qldb-driver-python.	20 août 2020
Pilote Amazon QLDB pour Go	Une version préliminaire du pilote Amazon QLDB pour Go est désormais disponible. Ce pilote vous permet d'utiliser leAWS SDK for Go pour interagir avec l'API de données transactionnelles de QLDB. Pour plus d'informations, consultez le pilote Amazon QLDB pour Go (version préliminaire).	6 août 2020
Livre de recettes de référence pour le pilote Python	Ajout d'une référence au livre de recettes qui présente des exemples de code de cas d'utilisation courants pour le pilote Amazon QLDB pour Python.	24 juillet 2020
Identifiants uniques dans Amazon QLDB	Ajout d'une nouvelle section qui décrit les propriétés et les directives d'utilisation des ID uniques dans Amazon QLDB.	9 juillet 2020

AWS CloudFormationressource pour les flux de journaux	Amazon QLDB prend désormais en charge une ressource permettant de créer des flux de journaux à l'aide d'unAWS CloudFormation modèle. Pour plus d'informations, consultez la AWS::QLDB::Streamressource dans le Guide deAWS CloudFormation l'utilisateur.	9 juillet 2020
Guide de référence du livre de recettes pour le pilote .NET	Ajout d'une référence au livre de recettes qui présente des exemples de code de cas d'utilisation courants pour le pilote Amazon QLDB pour .NET.	1er juillet 2020
Pilote Amazon QLDB pour .NET	Le pilote Amazon QLDB pour .NET est désormais disponible pour tous. Ce pilote est open source GitHub et vous permet de l'utiliserAWS SDK for .NET pour interagir avec l'API de données transactionnelles de QLDB. Pour les notes de publication, consultez le GitHub référentiel awslabs/amazon-qldb-driver-dotnet.	26 juin 2020

Pilote Amazon QLDB pour Node.js	Le pilote Amazon QLDB pour Node.js est désormais disponible pour tous. Ce pilote est open source GitHub et vous permet d'utiliser leAWS SDK de Node.js pour JavaScript interagir avec l'API de données transacti onnelles de QLDB. Pour les notes de publication, consultez leamazon-qldb-driver-nodejs GitHub référentiel awslabs/.	5 juin 2020
Flux de journaux Amazon QLDB	Amazon QLDB vous permet désormais de créer un flux qui capture chaque révision de document enregistrée dans votre journal et transmet ces données à Amazon Kinesis Data Streams en temps quasi réel. Pour plus d'informa tions, consultez Streaming de données de journaux depuis Amazon QLDB.	19 mai 2020
Exemples de code Amazon Ion	Ajout d'exemples de code qui interrogent et traitent les données Amazon Ion dans un registre Amazon QLDB à l'aide du pilote QLDB pour Java, Node.js et Python. Pour de plus amples informations, veuillez consulter Exemples de code Ion dans Amazon QLDB.	12 mai 2020

Modèle de simultanéité et contenu du journal	La rubrique sur le modèle de concurrence Amazon QLDB a été étendue afin d'ajouter des informations sur l'utilisation des index afin de limiter les conflits de contrôle optimiste de la concurrence (OCC). Ajout d'une nouvelle section qui décrit le contenu du journal dans Amazon QLDB.	4 mai 2020
Guide de démarrage rapide pour le pilote Node.js	Ajout d'un guide de démarrage rapide pour le pilote Amazon QLDB pour Node.js.	1er mai 2020
Coque Amazon QLDB	Le shell Amazon QLDB d'Amazon QLDB d'Amazon QLDB pour tous. Ce shell est open source GitHub et fournit une interface de ligne de commande qui vous permet d'exécuter des instructions PartiQL sur des données de registre. Pour les notes de publication, consultez leamazon-qldb-shell GitHub référentiel awslabs/.	le 20 avril 2020
Certifications de conformité	Amazon QLDB est désormais certifié pour les programme s deAWS conformité, notamment HIPAA et ISO. Pour plus d'informations, consultez Validation de conformité pour Amazon QLDB.	3 avril 2020

Recommandations étendues pour les conducteurs	La section des recommandations de pilotes Amazon QLDB a été étendue pour l'appliquer à tous les langages de programmation pris en charge.	2 avril 2020
Coque Amazon QLDB	Une version préliminaire du shell Amazon QLDB est désormais disponible. Ce shell fournit une interface de ligne de commande qui vous permet d'exécuter des instructions PartiQL sur des données de registre. Pour plus d'informations, consultez Accès à Amazon QLDB à l'aide du shell QLDB (API de données uniquement) (version préliminaire).	23 mars 2020
Pilote Java v1.1	Le pilote Amazon QLDB pour Java version 1.1 est désormais disponible pour tous. Pour les notes de publication, consultez leamazon-qldb-driver-java GitHub référentiel awslabs/.	le 20 mars 2020

Pilote Amazon QLDB pour .NET	Amazon QLDB propose désormais une version préliminaire du pilote .NET. Ce pilote vous permet d'utiliser leAWS SDK for .NET pour interagir avec l'API de données transactionnelles de QLDB. Pour plus d'informations, consultez le pilote Amazon QLDB pour .NET (version préliminaire).	13 mars 2020
Pilote Amazon QLDB pour Python	Le pilote Amazon QLDB pour Python est désormais généralement disponible. Ce pilote est open source GitHub et vous permet de l'utiliserAWS SDK for Python (Boto3) pour interagir avec l'API de données transactionnelles de QLDB. Pour les notes de publication, consultez leamazon-qldb-driver-python GitHub référentiel awslabs/.	11 mars 2020

UNDROP TABLEInstruction PartiQL et XML imbriqué	Dans Amazon QLDB, PartiQL prend désormais en charge les instructions DML (Data Manipulation Language) dans lesquelles les collections imbriquées sont spécifiées en tant que sources dans laFROM clause. Pour plus d'informations, consultez l'instruction FROM dans la référence PartiQL. QLDB PartiQL prend également en charge l'instruction UNDROP TABLE. Pour de plus amples informations, veuillez consulter Dédaction de tables.	26 février 2020
Rubrique d'introduction étendue	Extension de l'introduction Qu'est-ce qu'Amazon QLDB ? rubrique pour inclure les nouvelles sections Présentation d'Amazon QLDB et From Relational to Ledger.	24 janvier 2020
Référence PartiQL pour les fonctions prises en charge	La référence Amazon QLDB PartiQL a été étendue pour inclure des informations d'utilisation détaillées concernant les fonctions SQL prises en charge. Pour de plus amples informations, veuillez consulter Fonctions PartiQL.	2 janvier 2020

Recommandations pour les conducteurs et erreurs courantes	Ajout de nouvelles sections qui décrivent les recommandations de pilotes pour le pilote Amazon QLDB pour Java et les erreurs courantes pour toutes les langues du pilote.	2 janvier 2020
Lancement de nouvelles régions	Amazon QLDB est désormais disponible dans les régions Asie-Pacifique (Singapour), Asie-Pacifique (Singapour), Asie-Pacifique (Francfort). Pour obtenir la liste complète des régions disponibles, consultez les points de terminaison et les quotas Amazon QLDB dans le Référence générale d'Amazon Web Services.	19 novembre 2019
Pilote Amazon QLDB pour Node.js	Amazon QLDB propose désormais une version préliminaire du pilote Node.js. Ce pilote vous permet d'utiliser leAWS SDK de Node.js pour JavaScript interagir avec l'API de données transactionnelles de QLDB. Pour plus d'informations, consultez le pilote Amazon QLDB pour Node.js (version préliminaire).	13 novembre 2019

Pilote Amazon QLDB pour Python	Amazon QLDB propose désormais une version préliminaire du pilote Python. Ce pilote vous permet d'utiliser leAWS SDK for Python (Boto3) pour interagir avec l'API de données transactionnelles de QLDB. Pour plus d'informations, consultez le pilote Amazon QLDB pour Python (version préliminaire).	29 octobre 2019
Publication publique	Il s'agit de la première version publique d'Amazon QLDB. Cette version inclut le Guide du développeur et la référence de l'API de gestion intégrée des registres.	10 septembre 2019

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.